A APPENDIX

A.1 Sub-Algorithms of RPSP

A.1.1 submatrix Propagation. The **Algorithm 3 submatrix Propagation** conducts weighted sampling to generate $2^t \times 2^t$ submatrices based on the low-rankness of $2^{t-1} \times 2^{t-1}$ submatrices. The input of **Algorithm 3 submatrix Propagation** include the input matrix X, sampled $2^{t-1} \times 2^{t-1}$ submatrices, their singular values E^t, and the number of to be generated $2^t \times 2^t$ submatrices L_t.

Algorithm 3: submatrix Propagation

Inputs: $X, \mathcal{R}_{t-1}, E^{t-1}, L_t$
Outputs: $\mathcal{R}_t$
submatrix Propagation$(X, \mathcal{R}_{t-1}, E^{t-1}, L_t)$:
$\mathcal{R}_t \leftarrow \varnothing$
while $|R_t| < L_t$ **do**
  $Prob \sim U(0,1)$
  Randomly pick two non-overlapping submatrices from $\mathcal{R}_{t-1}$, denoted as $\mathcal{R}_{t-1}[i]$ and $\mathcal{R}_{t-1}[j]$
  $W \leftarrow \frac{E^{t-1}_{1,i} E^{t-1}_{1,j}}{\sum_{k=1}^{2^{t-1}} E^{t-1}_{k,i} \sum_{k=1}^{2^{t-1}} E^{t-1}_{k,j}}$
  if $Prob < W$ **then**
    $I \leftarrow$Row indices of $\mathcal{R}_{t-1}[i]$ and $\mathcal{R}_{t-1}[j]$
    $J \leftarrow$Column indices of $\mathcal{R}_{t-1}[i]$ and $\mathcal{R}_{t-1}[j]$
    $append(\mathcal{R}_t, X_{I \times J})$
end
return $\mathcal{R}_t$

A.1.2 Local Low Rank Prediction. RPSP further utilizes **Algorithm 4 Local Low Rank Prediction** to identify and reconstruct the local low rank matrices. RPSP computes a $M \times N$ scoring matrix S^T, in which S^T_{ij} stores the frequency of observing a large value of $\frac{\sigma_1}{||P||_*}$ among all the sampled $2^T \times 2^T$ submatrices that hits to X_{ij}. Hence, S^T_{ij} can be viewed as an approximation of the probability that X_{ij} is contained by a MLLRR submatrix with a size of $2^T \times 2^T$ or larger. Here we applied the **Spectral Co-Clustering** method developed by Dhillon et al [9] and the python library provided by scikit-learn [40] on S^T to identify local low rank submatrices. With the indices of each possible local low rank matrix identified, the local patterns were ranked by the level of their top singular values normalized by the sum of all singular values. Here the local patterns of the top K significant low rank property or with the top singular values large than a certain threshold form the final output of RPSP.

Algorithm 4: Local Low Rank Prediction

Inputs: $S^{T,M \times N}$
Outputs: $\mathcal{I} \times \mathcal{J}$
Local Low Rank Prediction$(S^{T,M \times N})$:
$\mathcal{I} \times \mathcal{J} \leftarrow$ **Spectral Co-Clustering**(S^T)
return $\mathcal{I} \times \mathcal{J}$

*A.1.3 Optimize the max pooling with respect to null space in **Algorithm 1**.* The max pooling with respect to null space can be further optimized by clustering the random unit vectors into groups of high cosine similarities. Specifically, the randomly sampled unit vectors were first clustered by using the K-mean of their cosine distance (1-cosine similarity). Then after $P_{(1)}, ..., P_{(r)}$ were identified, the null space of the linear span of $\{P_{(1)}, ..., P_{(r)}\}$ was estimated by the union of the clusters whose center P^C satisfies $\max\{\cos(P^C, P_{(1)}), ..., \cos(P^C, P_{(r)})\} < \cos(\theta)\}$. This approach effectively reduces the number of cosine distances needed to be computed.

A.1.4 Assessment of hyper-parameters of RPSP. RPSP has four hyper-parameters T, C, L_t, and N_t. L_t (number of randomly sampled or propagated submatrices) can be determined based on the input matrix size and computational capacity. T (number of layers for submatrix propagation) is set as 4 for efficient computation. C (threshold of $LowRankScore$) can be computed by randomly sampling $2^T \times 2^T$ submatrices of pure noise from randomly shuffled X and generating an empirical null distribution of $LowRankScore$. N_t (number of random unit vectors) can be determined by **Lemma 2**.

A.2 Mathematical Derivations And Considerations

A.2.1 Truncated SVD.. Let $X^{M \times N} = U \Sigma V^T (M \geq N)$ be the SVD of X, where $U^{M \times N}$ and $V^{N \times N}$ are left and right singular vector matrices, $\Sigma^{N \times N}$ is a diagonal matrix of singular values. Define $\Sigma^{(r)}$ such that $\Sigma^{(r)}_{ii} = \Sigma_{ii}, i \leq r; \Sigma^{(r)}_{ii} = 0, i > r$, i.e., only keeps the top r singular values > 0. The truncated SVD of X of rank r is defined as $tSVD(X, r) = U\Sigma^{(r)}V^T$. Noted, $Rank(X) \leq r$ if and only if $X = tSVD(X, r)$.

A.2.2 Mathematical considerations of the MLLRR problem. The biggest challenge with local low rank submatrix detection lies in that neither the row or column indices of the submatrix are known. As given in Definition 2, the low-rankness property of a submatrix is evaluated through the computation of its singular values, which apparently can't be evaluated until the submatrix has been presented. However, it is computationally impossible to go through all the submatrices of an input matrix. RPSP grows a submatrix of low rank from smaller ones, which utilizes two facts. Firstly, for a low rank matrix $X^{M \times N}$ of rank $r \ll min(M, N)$, the self consistency property suggests that any $M_0 \times N_0$ submatrix $(M_0, N_0 \geq r)$ randomly sample from X is most likely to have a rank of r (Lemma 1 in [37]). Secondly, for a given matrix, the total number of square submatrices of dimension M_0 grows exponentially with M_0. The first fact indicates that any submatrix of low rank is a collage or complete coverage of its own (smaller) submatrices, which are also of low rank. The second fact indicates that the only way for us to grow a local low rank submatrix is to start from the much smaller submatrices. In fact, for M_0 as small as 2, it is computationally feasible for us to obtain a full collection of $M_0 \times M_0$ submatrices that could densely cover $X^{M \times N}$. By teasing out all the $M_0 \times M_0$ submatrices of low rank, we could then gradually build them up into larger low rank submatrices. The evaluation of the low-rankness for a large number of submatrices now becomes computationally expensive. In RPSP, our biggest contribution is that we have developed a singular value approximation method using random projection to efficiently evaluate the low-rankness of any given submatrix, making it possible for us to build a submatrix from its parts.

A few examples can illustrate why a global search cannot effectively solve the MLLRR problem. We consider the following square matrices:

(1) $X^M \times M$ in which $X_{ij} \sim N(0,1)$ *i.i.d.*. Here X is a matrix of standard Gaussian error. The largest singular value of X is about $2\sqrt{M}$.
(2) $Y^M \times M$ in which $Y_{i,\cdot} \equiv Y'$, $Y'[i] \sim N(0,1)$ *i.i.d.*. Here Y is a matrix of rank=1 that has the same level of mean and standard deviation as X. Noted, the largest singular value of Y is about M.
(3) $Z^{M\times M}$ in which $Z_{ij} \equiv a$. The largest singular value of Y is $M\times a$.

Hence for a low rank sub-matrix of size $\sqrt{2M} \times \sqrt{2M}$ or smaller, whose mean and the standard deviation are not different from the background's, it is less likely to be identified by a global SVD as the largest singular value of the sub-matrix is about the same level of the largest singular value of the background noise matrix. However, if the low rank sub-matrix has a spiked mean, its largest singular value will be amplified by the spiked mean and the top singular vector of the whole matrix is naturally sparse. Hence, an LCV problem is more likely to be solved by a global search while the MLLRR problem of the insignificant mean difference between pattern and background is less likely to be detected by a global search. So, it is necessary to think of an alternative approach to solving the general MLLRR problem. Noted, the idea of screening a large set of small submatrices and propagating the low rank property of smaller ones to bigger submatrices only involves the computing of singular values of local patterns. Hence, we do not expect that the RPSP method may have disparate performances in solving LCV and LRR problems.

A.2.3 Mathematical formulations of SOTA methods. Co-clustering methods simultaneously cluster rows and columns of a two-dimensional data matrix. The general assumption is that the targeted submatrix has a larger or small mean value compared to the background noise. The Bregman co-clustering method generates a matrix partition I_k, J_k by preserving the maximum information of data X within the partitions. The approximation error $M(I,J) - M(\tilde{I},\tilde{J})$ represents the difference between the preserved information and original data, here $M(I,J)$ is the mutual information and $M(I,J) - M(\tilde{I} - \tilde{J}) = KL(dist_1(I,J)||dist_2(I,J))$. Laura et al. proposed the Plaid model to detect the submatrix by fitting each entry X_{ij} with K layers and make sure the summation of all layers $\sum_{k=1}^{K} \mu_k I_k J_k$ approximate the original value. Sparse SVD-based methods identify local low rank matrices by adding L1 sparse penalty to a global truncated SVD fitting. However, this type of method still demands distinct mean differences between pattern and background and trend to detect large low rank patterns that may explain the variance of the whole matrix. Lee et al. proposed the LLORMA method by using prior knowledge to select anchors of local low rank patterns. As listed in table 1, K_Ω^h is the kernel function with bandwidth h to smooth the projection value $P_\Omega(\cdot)$ near the anchor points Ω. However, this type of method, highly depends on prior knowledge that cannot solve the general MLLRR problem.

A.2.4 Proofs of Lemma 1 and Lemma 2. **Lemma 1 and 2** can be expressed and proved together. The two lemmas describe the following properties of random projection. For a given dimension R, denote $X^{R\times R}$ as an input matrix and $P \in \mathbb{R}^{R\times N^R}$ as a matrix of N^R randomly generated unit vectors in $\mathbb{R}^R$. $Y = XP$ denotes a random projection of X, then $\lim_{N^R\to\infty} \max_{1\le j\le N^R} \sqrt{\sum_{i=1}^R Y_{ij}^2} = \sigma_1$ and $\lim_{N^R\to\infty} \min_{1\le j\le N^R} \sqrt{\sum_{i=1}^R Y_{ij}^2} = \sigma_R$, here σ_1 and σ_R are the largest and smallest singular values of X. Denote $P_{(1)} = \arg\max_{P_{\cdot,j}} \sqrt{\sum_{i=1}^R Y_{ij}^2}$, $\lim_{N^R\to\infty} \max_{P_{\cdot,j}\in Sp(P_{(1)})^\perp} \sqrt{\sum_{i=1}^R Y_{ij}^2} = \sigma_2$, where $Sp(P_{(1)})^\perp$ denotes the null (or complemented) space of the linear space spanned by $P_{(1)}$. Similarly, define $P_{(r)} = \arg\max_{P_{\cdot,j}\in Sp(P_{(1)},\ldots,P_{(r-1)})^\perp} \sqrt{\sum_{i=1}^R Y_{ij}^2}$, $\lim_{N^R\to\infty} \max_{P_{\cdot,j}\in Sp(P_{(1)},\ldots,P_{(r-1)})^\perp} \sqrt{\sum_{i=1}^R Y_{ij}^2} = \sigma_r$, for $r \in 3, \ldots, R-1$.

Proof. Noted, $\sum_{i=1}^R Y_{ij}^2$ is the norm of the projection of X onto $Y_{\cdot,j}$. $Y_{\cdot,j} = XP_{\cdot,j} = U\Sigma V^T P_{\cdot,j} = \sum_{k=1}^R U_{\cdot,k}\Sigma_{kk}V_{k,\cdot}^T P_{\cdot,j}$, here $U\Sigma V^T$ is the SVD of X. Then we have $\sum_{i=1}^R Y_{ij}^2 = \sum_{i=1}^R (\sum_{k=1}^R U_{ik}\Sigma_{kk}V_{k,\cdot}^T P_{\cdot,j})^2 = \sum_{k=1}^R \sum_{i=1}^R (U_{ik})^2(\Sigma_{kk})^2(V_{k,\cdot}^T P_{\cdot,j})^2 = \sum_{k=1}^R (\Sigma_{kk})^2(V_{k,\cdot}^T P_{\cdot,j})^2$ as U is orthogonal, both Σ_{kk} and $V_{k,\cdot}^T P_{\cdot,j}$ are scalars. Hence the largest and smallest $\sum_{i=1}^R Y_{ij}^2$ is Σ_{11} and Σ_{RR}, which are achieved when $P_{\cdot,j}$ is $V_{1,\cdot}^T$ and $V_{R,\cdot}^T$, respectively. As $Sp(Y_{(1)}, \ldots, Y_{(r-1)})^\perp$ is the null space of the linear span of $Y_{(1)}, \ldots, Y_{(r-1)}$, the largest projection of X onto this space is σ_r when $P_{\cdot,j}$ is $V_{r,\cdot}^T$. □

A.3 Experimental Details

Detailed experimental parameters, data, and analysis are provided below. We conducted experiments on a GPU server of Cray HPE EX architecture featured with 64 nodes, 4 A100 GPUs, 64 cores, and 256GB per node and a CPU server features 640 compute nodes, each equipped with 256 GB of memory and two 64-core, 2.25 GHz, 225-watt AMD EPYC 7742 processors.

*A.3.1 Benchmark of **Algorithm 1: Singular Value Approximation**.* We have evaluated the computational efficiency and accuracy of **Algorithm 1** verses on conventional QR decomposition-based computation of singular values. We tested **Algorithm 1** on the GPU server and QR decomposition-based SVD on both the GPU server and a CPU server. Noted, due to the nature of QR decomposition, it is slower on GPU compared to CPU. We tested the two methods 50 times on 10^6 2×2, 10^5 4×4, and 10^5 8×8 matrices. The **Algorithm 1** used less than 10^{-4} second, which is on average about 10^5 faster than QR decomposition-based SVD. We also tested the methods 50 times on $10^5, 10^6, 10^7$ and 10^8 2×2 matrices and have observed the speed of **Algorithm 1** is consistently 10^5 faster than conventional SVD.

A.3.2 The Analysis of Sensitivity of the Half Angle θ. We have evaluated the sensitivity of the half angle θ in the RPSP on the synthetic data. We tested different θ values 10 times on these settings. we let $M = N = 1000$, pattern mean $\mu_k = \beta_k \times sd$ where $\beta_k = \{0, 0.2, 0.5\}$,

Table 2: Existing methods of MLLRR

Methods	Examples	Formulation	Tasks	Assumption
Co-clustering	Bregman; Plaid	$\min_{I_k, J_k, \mu_k} \sum_k \sum_{i \in I_k, j \in J_k} d(x_{ij}, \mu_k)$	LCV	Matrix partition
Matrix decomposition	SSVD; SPCA	$\min_{U,V}(\|\|X - UV^T\|\|_F^2 + \lambda_u \|\|U\|\|_1 + \lambda_v \|\|V\|\|_1),$	LCV MLLRR with LCV	Sparse patterns
Anchor based methods	LLORMA WEMAREC	$\min_{\hat{I}, \hat{J}, \hat{X}}(K_{X[\hat{I},\hat{J}]} \odot P_{X[\hat{I},\hat{J}]}(X - \hat{X}))$	MLLRR with LCV	submatrix detection

Table 3: The Singular Value Approximation and SVD Running Time

q=2,nc=5	Inner time(gpu)	Inner time(cpu)	Svd time(cpu)
ns=1e4	0.00002	0.0003	0.9
ns=1e5	0.00002	0.0031	0.1166
ns=1e6	0.000036	0.0328	1.1563
ns=1e7	0.0002	0.3664	11.6114
ns=1e8	0.037	3.6	116
q=4,nc=27	**Inner time(gpu)**	**Inner time(cpu)**	**Svd time(cpu)**
ns=1e4	0.000027	0.00235	0.0325
ns=1e5	0.000028	0.02436	0.3077
q=8,nc=767	**Inner time(gpu)**	**Inner time(cpu)**	**Svd time(cpu)**
ns=1e4	0.000032	0.0959	0.0828
ns=1e5	0.000059	0.9422	0.8171
q=16,nc=588804	**Inner time(gpu)**	**Inner time(cpu)**	**Svd time(cpu)**
ns=500	0.000349	NA	NA
q=16,nc=3544	**Inner time(gpu)**	**Inner time(cpu)**	**Svd time(cpu)**
ns=1000	0.00003	0.1071	0.0268
ns=10000	0.000056	1.07	0.2542

Table 4: Half Angle θ and The Number of Unit Vectors

5		10		15		20	
R	N^R	R	N^R	R	N^R	R	N^R
2	132	2	34	2	15	2	9
4	1100	4	1100	4	223	4	74
8	2000	8	2000	8	2000	8	2000
16	10000	16	10000	16	10000	16	10000

25		30		35		40	
R	N^R	R	N^R	R	N^R	R	N^R
2	6	2	5	2	4	2	3
4	32	4	17	4	10	4	4
8	983	8	257	8	86	8	35
16	10000	16	10000	16	7287	16	1178

relative noise level $\alpha_k = 0, 0.2, 0.5$, pattern size $m_k = n_k = \{200, 500\}$, half angle $\theta = \{5, 10, 15, 20, 25, 30, 35, 40\}$ *degree*. For the settings for R and N^R please see table 4. And we evaluated the sensitivity of θ by F1-Score. For the performance, please see Table 5 and Table 6.

A.3.3 *Parameter settings of RPSP.*

In this study, we set $T = 4$ and have validated that this setting can accurately identify MLLRR submatrices in different scenarios. In this study, we set N_1 =40, N_2 =200, N_3 =2000, and N_4 =6000 that guarantee almost surely that the max cosine distance between any 2, 4, 8, and 16 dimension vector and the random unit vectors is larger than 0.98, 0.95, 0.92 and 0.8, respectively. We set the initial L_1=10^7, $L_t = \frac{L_{t-1}}{10}$, and we also accumulated the scoring matrix S^t by adding S^{t-1}, and forgetting some scores with a random probability. This accumulation strategy can help our model learn from the previous iterations, and thus drastically reduce the running time and increase accuracy.

A.3.4 *Parameter settings of baseline methods.*

RPSP is implemented by python 3.9.8 version and the python libraries numpy(1.19.0)[18], pandas(1.4.1)[38], pytorch(1.6.0)[39],numba(0.50.1)[24] etc.

The baseline methods includes SSVD [57], SPCA [13], Plaid [25], CC [2], LLORMA [26]. The first four methods were implemented in R environment. For SSVD, we used R package ssvd (version 1.0) and default parameters. For SPCA, we used R package sparsepca (version 0.1.2) and default parameters. For two bicluster methods Plaid and CC, we used the R package biclust (version 2.0.1). Specifically for Plaid, we set the 'background' parameter as True, 'fit.model' parameter as $y \sim m + a + b$ by following the tutorial. And for CC method, the parameter δ and α was set as 1.0 and 1.5 as instructed by the tutorial.

For LLORMA, we used the Global LLORMA from the author's GitHub[1] with library Tensorflow-GPU 1.4.0. For synthetic data experiments, all experiments were conducted by setting PRE_RANK=1 along with other default parameters. In real-world data experiments, we set PRE_RANK = 10 along with other parameters in default. We used the default learning rate parameter for all cases except for when testing the time consumption on input matrices of sizes 10000×10000 and 5000×5000, where the learning rate is set to be PRE_LARNING_RATE = 2e-5.

A.3.5 *Evaluation metric of synthetic data-based experiments.*

For the simulated data. The Accuracy of detecting the true pattern and the background noise is used in our experiments to evaluate the performance of RPSP and the benchmarks. For one simulated input matrix $X^{M \times N}$, we labeled the true pattern as "1" and the background noise as "0". We keep tracking the index of the true pattern, thus we could generate a binary matrix as the ground

[1] https://github.com/JoonyoungYi/LLORMA-tensorflow

truth $GT^{M\times N}$. If we define the output from the methods above to be $X_{output}^{M\times N}$ and change the non-zero elements in $X_{output}^{M\times N}$ to "1". We could get the output binary matrix $GT_{output}^{M\times N}$. Then we can get the criterion of True Positive(TP), True Negative(TN), False Positive(FP), False Negative(FP) and the Accuracy by the following function:

$$TP, FP, FN, TN \leftarrow Confusion\_Matrix(GT_{output}^{M\times N}, GT^{M\times N}) \tag{A.1}$$

$$Accuracy \leftarrow \frac{TP + TN}{TP + TN + FP + FN} \tag{A.2}$$

Note that in some simulation settings, the method SPCA failed, so we gave its Accuracy of 0. In fact, all "0" Accuracy in the figures is because the methods could fail to give an output. For method CC, it sometimes detects the whole matrix as one pattern matrix, meaning that CC could not identify any of the true patterns. Under our evaluation metrics, CC will result in TP=1, TN=0, FP=0, FN=1, so we gave the Accuracy of 0.5.

A.3.6 Real-World data processing. **MovieLens data:** We have gotten permission from GroupLens to use the MovieLens dataset[17] in our experiments and the datasets don't include sensitive information. MovieLens 25M data contains the ratings of 62,000 movies by 162,000 users. The MovieLens data is commonly used as benchmark data for pattern detection. To ensure the rigor of evaluation, we selected the top 600 active users(rows) and the 600 most rated movies(columns) to build a testing dataset with a density rate of 20.0%. This is a relatively sparse dataset. Indeed, on high-sparsity datasets, all baseline methods tend to identify LCV submatrices, but still, RPSP is more favorable than others in terms of the low-rankness and coverage rate of the detected patterns.

Single Cell RNA-sequencing data: Single-cell RNA-sequencing (scRNA-seq) is a high throughput technique that measures the gene expression profile of individual cells [41, 49]. The researchers are allowed to use the dataset in their study and the datasets don't include sensitive information. The real application was performed on two biomedical datasets, which are melanoma and head and neck cancer scRNA-seq data. We collected these two datasets from Gene Expression Omnibus (GEO) database, with accession ID GSE72056 and GSE103322. The cell type label and sample information provided in the original work was directly utilized. The GSE72056 data is collected on human melanoma tissues. The original paper provided cell classification and annotations including B cells, cancer-associated fibroblast (CAF) cells, endothelial cells, macrophage cells, malignant cells, NK cells, T cells, and unknown cells. The GSE103322 data is collected on head and neck cancer tissues. The original paper provided cell classification and annotations including B cells, dendritic cells, endothelial cells, fibroblast cells, macrophage cells, malignant cells, mast cells, myocyte cells, and T cells. We utilized a standard normalization protocol (FPKM) for both datasets, and selected the 4000 genes (rows) and 2000 cells (columns) with the highest expression values, bringing the density rates to 70.14% (GSE72056) and 75.71% (GSE103322). Notably, as indicated by the original work, malignant cells have high intertumoral heterogeneity. These two datasets provide us a great opportunity to analyze the biological mechanism by identifying the local low rank pattern within the data. And the total citation of the two works is above 2000.

Both data sets have been utilized in more than 100 studies, in which GSE72056 contains 23684 genes and 4486 cell, and GSE103322 contains 22494 genes and 5902 cells. We utilize the standardized TPM measure of gene expression level as the input. Firstly, we first conducted a standardized normalization of the data by taking log+1: $X \leftarrow log_2(X+1)$. We further selected the 4000 genes (rows) of the top averaged expression level and the 2000 cells (columns) of the top total expression level to build our input testing data $X^{4000\times 2000}$. The low-rankness, Coverage Size(Size), Coverage Rate, and Running Time were used as metrics in our experiments to evaluate the performance of RPSP and benchmark with other methods. For each submatrix calculated by the methods above, we compute its COR rate, Size(how many elements in it), and the coverage of rate of the submatrix to the input real data.

The rows represent the genes and the columns represent the cell. Each element in the matrix means gene expression. The scRNA-seq data is sparse, the zero value in the matrix means the gene is not expressed in the cell. Thus, we just select 4000 rows and 2000 columns by their top mean value in our experiments. Specifically, the scRNA-seq data is the unstructured data, it isn't like the image data, and the order of rows or columns doesn't have a special mean. In the experiments on scRNA-seq data, our goal is to get the local low rank submatrix from these data. The results (Fig S1) show that RPSP performances are great to get the submatrices of MLLRR structure. The figures of Coverage Rate and Size show that RPSP can get the most obvious MLLRR structure submatrix and coverage enough submatrix size(product of a number of rows and columns). The Running Time shows that RPSP has good time performance. RPSP consistently identified the local rank-1 pattern under this experimental setting.

Spatial transcriptomics data: 10x Genomics spatial transcriptomics (ST) is a recent commercialized technique to measure spatial coordinates associated with gene expression signals from a biological tissue sample, and it has a huge utilization in biomedical studies. The researchers are allowed to use the dataset in their study and the datasets don't include personal information. We collected the spatial transcriptomics data on human breast cancer tissue (v1.1 section 1) from https://www.10xgenomics.com/resources/datasets/, consisting of 13161 genes and 3798 spatial spots. The data was processed and visualized by using Seurat 4.0 R package[16]. Counts data were directly utilized as the input of RPSP and other baseline methods. The row represents the genes and the column represents the cell. our goal is to get the local low rank submatrix from this data. The result(Fig S1e) shows one case of local low rank submatrix found by RPSP.

A.3.7 Evaluation metrics of real-world data. Four evaluation metrics were utilized to evaluate the performance of each method, namely (1) the low-rankness, evaluated by $\frac{\sigma_1}{||X_{I_k\times J_k}||_*}$ of an identified local low rank matrix $X_{I_k\times J_k}$: $\sum_{i\in I_k} PCC(X_{i,J_k}, V_1)$, where σ_1 is the largest singular value and $||X_{I_k\times J_k}||_*$ is the nuclear norm of $X_{I_k\times J_k}$, (2) the Size of a local low rank matrix, (3) the total Coverage Rate defined as the total number of entries in the top-k identified patterns divided by the size of the input matrix, and (4) the Running Time. In addition, the context-specific meaning of the identified patterns was evaluated by prior knowledge of the row/column-wise features.

A.3.8 Comprehensive summary of real-world data-based experiments. **Application on movieLens data:** MovieLens 25M data contains the ratings of 62,000 movies by 162,000 users provided by GroupLens[17]. The MovieLens data is commonly used as benchmark data for pattern detection. To ensure the rigor of evaluation, we selected the top 600 active users(rows) and the 600 most rated movies(columns) to build a testing dataset with a density rate of 20.0%. This is a relatively sparse dataset. Indeed, on high-sparsity datasets, all baseline methods tend to identify LCV submatrices, but still, RPSP is more favorable than others in terms of the low-rankness, and coverage rate.

We evaluated the top significant local low rank matrices identified by each method. The low-rankness of the top six significant local low rank matrices identified RPSP is consistently higher (~0.8) than the ones detected by baseline methods (~0.55) (**Fig S1a**), while the size of the patterns detected by RPSP is lower but at a similar level of the ones detected by LLORMA, SSVD, and SPCA **Fig S1b**). Although the patterns detected by RPSP is slightly smaller, their coverage rate is higher than the results of LLORMA, SSVD and SPCA. This is because the ones detected by RPSP are distinctly non-overlapping while the top patterns identified by baseline methods are heavily overlapped (**Fig S1c**). On this data, Plaid did not detect any pattern while CC detected a large pattern formed by 309 users and 221 movies, whose low-rankness is lower than the ones detected by RPSP. The RPSP has a longer running time than LLORMA, SSVD, and SPCA but is faster than CC. In sum, on the MovieLens data, FFLRM could detect distinct local low rank matrices of specifically high low-rankness.

Application on single-cell RNA-seq data: Single-cell RNA-sequencing (scRNA-seq) is a high throughput technique commonly used in studying complex biological systems [41, 49]. A typical scRNA-seq data is a matrix of ~10,000 genes (rows) in ~5,000 individual cells (columns) with a density rate in the range of 5% – 50%. The LCV and MLLRR submatrices in scRNA-seq data directly correspond to the subpopulation of cells with distinct functions [51]. We applied RPSP and SOTA methods on two real-world scRNA-seq data that are most commonly utilized in testing pattern detection methods, namely GSE72056 (melanoma) and GSE103322 (head and neck cancer) [41, 49]. Due to the page limit, we only presented the results on GSE103322 in the main text. But here, we are presenting the results for both scRNA-Seq datasets. We utilized a standard normalization protocol (FPKM) and built the testing data by selecting the 4000 genes (rows) and 2000 cells (columns) with the highest expression values, bringing the density rates to 70.14% (GSE72056) and 75.71% (GSE103322).

On the two scRNA-seq data, both the low-rankness and the size of the top patterns identified by RPSP are consistently higher than the ones detected by baseline methods (**Fig S1a-b**). The patterns detected by RPSP are much less overlapped than the ones detected by LLORMA and SSVD. RPSP also achieved the highest total coverage rate compared to all baseline methods (**Fig S1c**). Plaid and CC only detected one local pattern, whose low-rankness is much lower than the ones detected by RPSP, LLORMA, and SSVD. The RPSP had a longer running time than SSVD and SPCA but is faster than LLORMA and CC on the scRNA-seq data (**Fig S1c**). We also examined the biological meaning of the low rank patterns detected by RPSP, LLORMA and CC, by testing the enrichment of the gene features of each pattern against known biological pathways. Noted, only in RPSP, we found three local low rank matrices significantly enrich distinct biological functions including cell metabolism, cell proliferation, and antigen presentation. In summary, RPSP outperforms all baseline methods in detecting local low rank matrices on the scRNA-seq data sets, in terms of the low-rankness, size, coverage rate, and biological interpretability of detected patterns.

Application on spatial transcriptomic data: 10x Genomics spatial transcriptomics (ST) is a recent commercialized technique to measure spatial coordinates associated with gene expression signal from a biological tissue sample, and it has been widely utilized in biomedical studies. A typical ST data is a matrix of ~15,000 genes (rows) in ~4,000 individual spatial spots (columns), and each spot has a 2D spatial coordinate (**Fig S1d**). A key challenge in ST data analysis is to infer the spatially dependent biological functional variations, which could be modeled as local low rank matrices formed by functionally associated genes over a certain spatial region, i.e. the MLLRR problem.

We applied RPSP and SOTA methods on the v1.1 ST data of breast cancer provided by 10xgenomics.com, consisting of 13161 genes and 3798 spatial spots with a density rate of 40.56%. Noted, as we have seen in the MovieLens and scRNA-seq data, RPSP is the only method that detected patterns of strong low-rankness. We showcased one of the MLLRR patterns specifically detected by RPSP (**Fig S1e**). The genes of this submatrix include two major groups, MHC class-I (immune signal given from cancers) and MHC class-II (immune signal received by immunes) antigen-presenting genes. The spatial coordinates and signal level of this submatrix suggested the region of different levels of immune response in the cancer tissue (red regions in **Fig S1f** are of the high immune response).

REFERENCES

[1] Dimitris Achlioptas. 2003. Database-friendly random projections: Johnson-Lindenstrauss with binary coins. *Journal of computer and System Sciences* 66, 4 (2003), 671–687.

[2] Arindam Banerjee, Inderjit Dhillon, Joydeep Ghosh, Srujana Merugu, and Dharmendra S Modha. 2007. A generalized maximum entropy approach to bregman co-clustering and matrix approximation. *Journal of Machine Learning Research* 8, Aug (2007), 1919–1986.

[3] Ella Bingham and Heikki Mannila. 2001. Random projection in dimensionality reduction: applications to image and text data. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining.*

[4] Thierry Bouwmans and El Hadi Zahzah. 2014. Robust PCA via principal component pursuit: A review for a comparative evaluation in video surveillance. *Computer Vision and Image Understanding* 122 (2014), 22–34.

[5] Wennan Chang, Changlin Wan, Yong Zang, Chi Zhang, and Sha Cao. 2020. Supervised clustering of high dimensional data using regularized mixture modeling. *arXiv preprint arXiv:2007.09720* (2020).

[6] Chao Chen, Dongsheng Li, Yingying Zhao, Qin Lv, and Li Shang. 2015. WEMAREC: Accurate and scalable recommendation through weighted and ensemble matrix approximation. In *Proceedings of the 38th ACM SIGIR*. 303–312.

[7] Yao Cheng, Liang Yin, and Yong Yu. 2014. LorSLIM: low rank sparse linear methods for top-n recommendations. In *2014 IEEE International Conference on Data Mining*. IEEE, 90–99.

[8] Biswa Nath Datta. 2010. *Numerical linear algebra and applications*. Vol. 116. Siam.

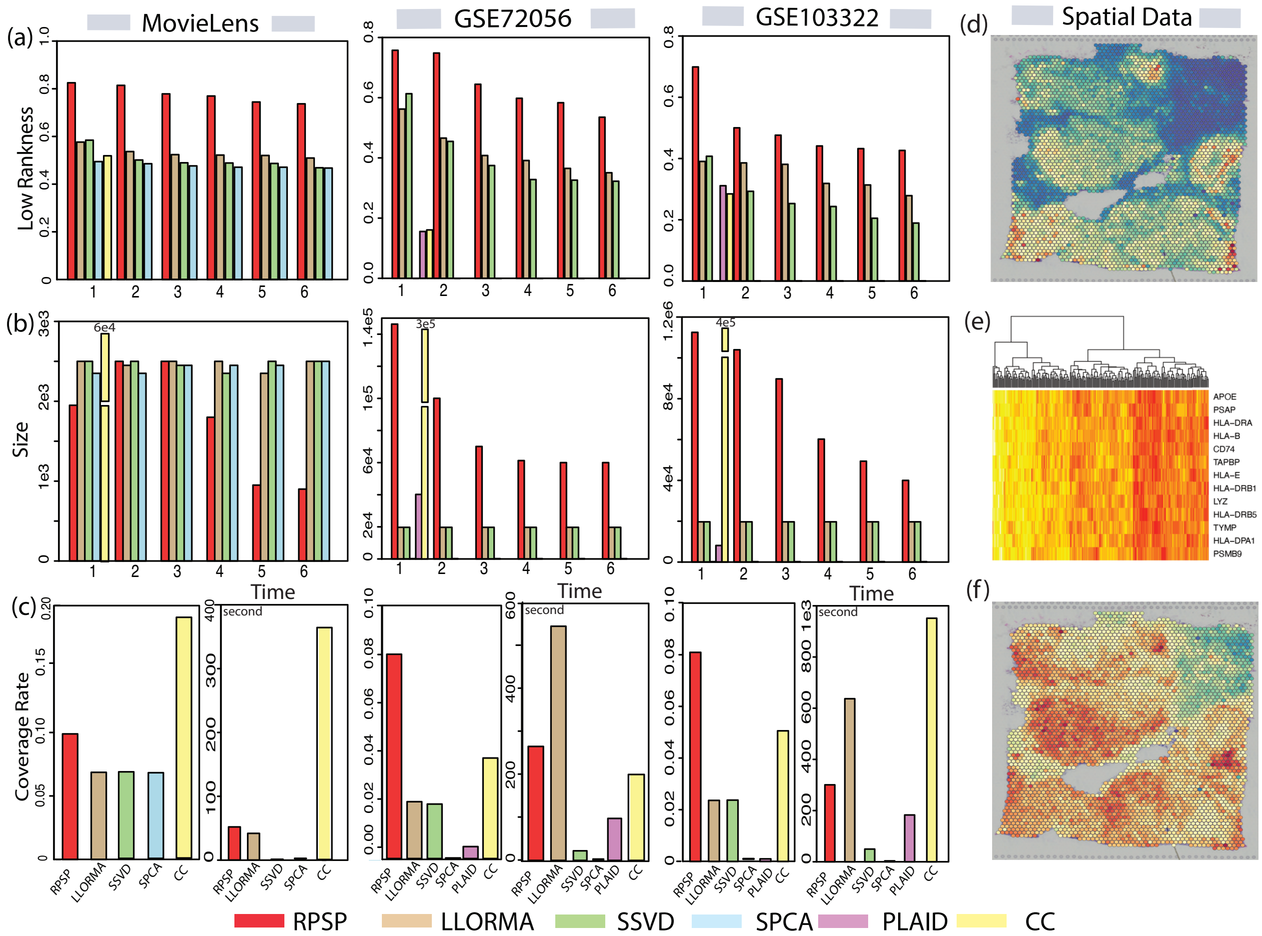

Fig S1. Experiment on real-world data.

[9] Inderjit S Dhillon. 2001. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. 269–274.

[10] Chris Ding, Tao Li, Wei Peng, and Haesun Park. 2006. Orthogonal nonnegative matrix t-factorizations for clustering. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. 126–135.

[11] Eleni Drinea, Petros Drineas, and Patrick Huggins. 2001. A randomized singular value decomposition algorithm for image processing applications. In *Proceedings of the 8th panhellenic conference on informatics*. Citeseer, 278–288.

[12] Petros Drineas and Michael W Mahoney. 2016. RandNLA: randomized numerical linear algebra. *Commun. ACM* 59, 6 (2016), 80–90.

[13] N Benjamin Erichson, Peng Zheng, Krithika Manohar, Steven L Brunton, J Nathan Kutz, and Aleksandr Y Aravkin. 2020. Sparse principal component analysis via variable projection. *SIAM J. Appl. Math.* 80, 2 (2020), 977–1002.

[14] Gene H Golub and Charles F Van Loan. 2013. *Matrix computations*. JHU press.

[15] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. 2011. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review* 53, 2 (2011), 217–288.

[16] Yuhan Hao, Stephanie Hao, Erica Andersen-Nissen, William M. Mauck III, Shiwei Zheng, Andrew Butler, Maddie J. Lee, Aaron J. Wilk, Charlotte Darby, Michael Zagar, Paul Hoffman, Marlon Stoeckius, Efthymia Papalexi, Eleni P. Mimitou, Jaison Jain, Avi Srivastava, Tim Stuart, Lamar B. Fleming, Bertrand Yeung, Angela J. Rogers, Juliana M. McElrath, Catherine A. Blish, Raphael Gottardo, Peter Smibert, and Rahul Satija. 2021. Integrated analysis of multimodal single-cell data. *Cell* (2021). https://doi.org/10.1016/j.cell.2021.04.048

[17] F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)* 5, 4 (2015).

[18] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. 2020. Array programming with NumPy. *Nature* 585, 7825 (Sept. 2020), 357–362. https://doi.org/10.1038/s41586-020-2649-2

[19] Sepp Hochreiter, Ulrich Bodenhofer, Martin Heusel, Andreas Mayr, Andreas Mitterecker, Adetayo Kasim, Tatsiana Khamiakova, Suzy Van Sanden, Dan Lin, Willem Talloen, et al. 2010. FABIA: factor analysis for bicluster acquisition. *Bioinformatics* 26, 12 (2010), 1520–1527.

[20] Wenpin Hou, Zhicheng Ji, Hongkai Ji, and Stephanie C Hicks. 2020. A Systematic Evaluation of Single-cell RNA-sequencing Imputation Methods. *bioRxiv* (2020).

[21] Zhanxuan Hu, Feiping Nie, Rong Wang, and Xuelong Li. 2021. Low rank regularization: A review. *Neural Networks* 136 (2021), 218–232.

[22] Da Wei Huang, Brad T Sherman, Qina Tan, Joseph Kir, David Liu, David Bryant, Yongjian Guo, Robert Stephens, Michael W Baseler, H Clifford Lane, et al. 2007. DAVID Bioinformatics Resources: expanded annotation database and novel algorithms to better extract biology from large gene lists. *Nucleic acids research* 35, suppl_2 (2007), W169–W175.

[23] N Kishore Kumar and Jan Schneider. 2017. Literature survey on low rank approximation of matrices. *Linear and Multilinear Algebra* 65, 11 (2017), 2212–2244.

[24] Siu Kwan Lam, Antoine Pitrou, and Stanley Seibert. 2015. Numba: A llvm-based python jit compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*. 1–6.

[25] Laura Lazzeroni and Art Owen. 2002. Plaid models for gene expression data. *Statistica sinica* (2002), 61–86.

[26] Joonseok Lee, Seungyeon Kim, Guy Lebanon, Yoram Singer, and Samy Bengio. 2016. LLORMA: Local low-rank matrix approximation. *JMLR* 17 (2016), 442–465.

[27] Mihee Lee, Haipeng Shen, Jianhua Z Huang, and JS Marron. 2010. Biclustering via sparse singular value decomposition. *Biometrics* 66, 4 (2010), 1087–1095.

[28] Jingjing Li, Ke Lu, Zi Huang, and Heng Tao Shen. 2019. On both cold-start and long-tail recommendation with social data. *IEEE Transactions on Knowledge and Data Engineering* 33, 1 (2019), 194–208.

[29] Mu Li, Wei Bi, James T Kwok, and Bao-Liang Lu. 2014. Large-scale Nyström kernel matrix approximation using randomized SVD. *IEEE transactions on neural networks and learning systems* 26, 1 (2014), 152–164.

[30] Edo Liberty, Franco Woolfe, Per-Gunnar Martinsson, Vladimir Rokhlin, and Mark Tygert. 2007. Randomized algorithms for the low-rank approximation of matrices. *Proceedings of the National Academy of Sciences* 104, 51 (2007), 20167–20172.

[31] Huafeng Liu, Liping Jing, Yuhua Qian, and Jian Yu. 2019. Adaptive local low-rank matrix approximation for recommendation. *ACM Transactions on Information Systems (TOIS)* 37, 4 (2019), 1–34.

[32] Michael W Mahoney et al. 2011. Randomized algorithms for matrices and data. *Foundations and Trends® in Machine Learning* 3, 2 (2011), 123–224.

[33] Ivan Markovsky. 2008. Structured low-rank approximation and its applications. *Automatica* 44, 4 (2008), 891–909.

[34] Per-Gunnar Martinsson, Vladimir Rokhlin, and Mark Tygert. 2011. A randomized algorithm for the decomposition of matrices. *Applied and Computational Harmonic Analysis* 30, 1 (2011), 47–68.

[35] Per-Gunnar Martinsson and JA Tropp. 2020. Randomized numerical linear algebra: foundations & algorithms (2020). *arXiv preprint arXiv:2002.01387* (2020).

[36] Bamdev Mishra, Gilles Meyer, Francis Bach, and Rodolphe Sepulchre. 2013. Low-rank optimization with trace norm penalty. *SIAM Journal on Optimization* 23, 4 (2013), 2124–2149.

[37] Art B Owen and Patrick O Perry. 2009. Bi-cross-validation of the SVD and the nonnegative matrix factorization. (2009).

[38] The pandas development team. 2020. *pandas-dev/pandas: Pandas*. https://doi.org/10.5281/zenodo.3509134

[39] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[40] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[41] Sidharth V Puram, Itay Tirosh, Anuraag S Parikh, Anoop P Patel, et al. 2017. Single-cell transcriptomic analysis of primary and metastatic tumor ecosystems in head and neck cancer. *Cell* 171, 7 (2017), 1611–1624.

[42] Benjamin J Sapp and Written Preliminary Examination II. 2011. Randomized algorithms for low rank matrix decomposition. *Computer and Information Science, University of Pennsylvania* 24 (2011).

[43] Tamas Sarlos. 2006. Improved approximation algorithms for large matrices via random projections. In *2006 47th annual IEEE symposium on foundations of computer science (FOCS'06)*. IEEE, 143–152.

[44] Romain Serizel, Marc Moonen, Bas Van Dijk, and Jan Wouters. 2014. Low-rank approximation based multichannel Wiener filter algorithms for noise reduction with application in cochlear implants. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 22, 4 (2014), 785–799.

[45] Moein Shakeri and Hong Zhang. 2016. COROLA: A sequential solution to moving object detection using low-rank approximation. *Computer Vision and Image Understanding* 146 (2016), 27–39.

[46] Haipeng Shen and Jianhua Z Huang. 2008. Sparse principal component analysis via regularized low rank matrix approximation. *JMA* 99, 6 (2008), 1015–1034.

[47] Martin Sill, Sebastian Kaiser, Axel Benner, and Annette Kopp-Schneider. 2011. Robust biclustering by sparse singular value decomposition incorporating stability selection. *Bioinformatics* 27, 15 (2011), 2089–2097.

[48] Larry J Stockmeyer. 1975. *The set basis problem is NP-complete.* IBM Thomas J. Watson Research Division.

[49] Itay Tirosh, Benjamin Izar, Sanjay M Prakadan, Marc H Wadsworth, et al. 2016. Dissecting the multicellular ecosystem of metastatic melanoma by single-cell RNA-seq. *Science* 352, 6282 (2016), 189–196.

[50] Aaron R Voelker, Jan Gosmann, and Terrence C Stewart. 2017. Efficiently sampling vectors and coordinates from the n-sphere and n-ball. *Centre for Theoretical Neuroscience-Technical Report* 1 (2017).

[51] Changlin Wan, Wennan Chang, Yu Zhang, Fenil Shah, Xiaoyu Lu, Yong Zang, Anru Zhang, Sha Cao, Melissa L Fishel, Qin Ma, et al. 2019. LTMG: a novel statistical modeling of transcriptional expression states in single-cell RNA-Seq data. *Nucleic acids research* 47, 18 (2019), e111–e111.

[52] Daniela M Witten, Robert Tibshirani, and Trevor Hastie. 2009. A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis. *Biostatistics* 10, 3 (2009), 515–534.

[53] Svante Wold. 1978. Cross-validatory estimation of the number of components in factor and principal components models. *Technometrics* 20, 4 (1978), 397–405.

[54] Aaron D Wyner. 1967. Random packings and coverings of the unit n-sphere. *The Bell System Technical Journal* 46, 9 (1967), 2111–2118.

[55] Chun-Qiu Xia, Ke Han, Yong Qi, Yang Zhang, and Dong-Jun Yu. 2017. A self-training subspace clustering algorithm under low-rank representation for cancer classification on gene expression data. *IEEE/ACM transactions on computational biology and bioinformatics* 15, 4 (2017), 1315–1324.

[56] Yang Xu, Lei Zhu, Zhiyong Cheng, Jingjing Li, and Jiande Sun. 2020. Multi-feature discrete collaborative filtering for fast cold-start recommendation. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 270–278.

[57] Dan Yang, Zongming Ma, and Andreas Buja. 2014. A sparse singular value decomposition method for high-dimensional data. *Journal of Computational and Graphical Statistics* 23, 4 (2014), 923–942.

[58] Xuejiao Yang and Bang Wang. 2019. Local matrix approximation based on graph random walk. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1037–1040.

[59] Hui Zou, Trevor Hastie, and Robert Tibshirani. 2006. Sparse principal component analysis. *Journal of computational and graphical statistics* 15, 2 (2006), 265–286.

Table 5: Sensitivity of Half Angle θ When Pattern Size=200

M=N=1000, Pattern Size=200											
Θ	cos(2Θ)	Error Shift	Mean Shift	Time	F1-Score	Θ	cos(2Θ)	Error Shift	Mean Shift	Time	F1-Score
			0.0	536.520	0.784				0.0	490.890	0.834
		0.0	0.2	535.960	0.804			0.0	0.2	486.380	0.898
			0.5	518.793	0.837				0.5	491.440	0.928
			0.0	531.470	0.771				0.0	497.453	0.744
5	0.98	0.2	0.2	518.477	0.765	25	0.64	0.2	0.2	499.897	0.787
			0.5	535.590	0.768				0.5	490.557	0.797
			0.0	532.390	0.719				0.0	487.947	0.702
		0.5	0.2	536.567	0.730			0.5	0.2	491.737	0.711
			0.5	521.370	0.734				0.5	495.123	0.711
			0.0	507.053	0.777				0.0	494.383	0.769
		0.0	0.2	517.963	0.833			0.0	0.2	497.333	0.840
			0.5	507.117	0.869				0.5	488.317	0.840
			0.0	505.473	0.776				0.0	494.963	0.701
10	0.94	0.2	0.2	502.483	0.806	30	0.5	0.2	0.2	497.177	0.767
			0.5	498.667	0.820				0.5	486.790	0.790
			0.0	502.450	0.689				0.0	492.087	0.714
		0.5	0.2	504.947	0.733			0.5	0.2	494.787	0.710
			0.5	504.397	0.731				0.5	486.430	0.701
			0.0	494.650	0.843				0.0	482.707	0.744
		0.0	0.2	505.940	0.865			0.0	0.2	480.140	0.807
			0.5	493.570	0.882				0.5	481.660	0.793
			0.0	494.777	0.767				0.0	478.973	0.644
15	0.87	0.2	0.2	494.540	0.750	35	0.34	0.2	0.2	492.917	0.790
			0.5	492.507	0.861				0.5	483.210	0.791
			0.0	497.627	0.699				0.0	487.500	0.667
		0.5	0.2	494.030	0.681			0.5	0.2	484.487	0.666
			0.5	504.290	0.713				0.5	488.207	0.694
			0.0	492.210	0.787				0.0	480.470	0.762
		0.0	0.2	494.873	0.820			0.0	0.2	479.880	0.765
			0.5	495.190	0.826				0.5	482.103	0.818
			0.0	497.407	0.806				0.0	471.613	0.713
20	0.77	0.2	0.2	503.627	0.838	40	0.17	0.2	0.2	483.130	0.732
			0.5	496.477	0.830				0.5	474.973	0.738
			0.0	494.847	0.664				0.0	480.857	0.639
		0.5	0.2	496.273	0.677			0.5	0.2	476.760	0.657
			0.5	493.117	0.692				0.5	479.040	0.724

Table 6: Sensitivity of Half Angle θ when Pattern Size=500

M=N=1000, Pattern Size=500											
Θ	cos(2Θ)	Error Shift	Mean Shift	Time	F1-Score	Θ	cos(2Θ)	Error Shift	Mean Shift	Time	F1-Score
			0.0	528.080	1.000				0.0	488.047	0.988
		0.0	0.2	513.907	1.000			0.0	0.2	511.963	0.982
			0.5	521.830	1.000				0.5	487.417	0.997
			0.0	526.200	1.000				0.0	497.473	0.948
5	0.98	0.2	0.2	518.637	1.000	25	0.64	0.2	0.2	506.750	0.963
			0.5	530.297	1.000				0.5	526.657	0.968
			0.0	525.837	0.936				0.0	498.527	0.901
		0.5	0.2	527.670	0.945			0.5	0.2	507.567	0.909
			0.5	546.290	0.934				0.5	498.077	0.968
			0.0	506.880	1.000				0.0	502.963	0.969
		0.0	0.2	504.187	1.000			0.0	0.2	494.053	0.985
			0.5	519.347	1.000				0.5	493.323	0.989
			0.0	506.650	0.990				0.0	515.800	0.931
10	0.94	0.2	0.2	506.533	0.995	30	0.5	0.2	0.2	498.913	0.979
			0.5	511.723	0.978				0.5	503.250	0.979
			0.0	506.970	0.921				0.0	498.720	0.929
		0.5	0.2	503.280	0.936			0.5	0.2	509.713	0.930
			0.5	523.963	0.948				0.5	524.183	0.941
			0.0	509.060	0.999				0.0	491.587	0.972
		0.0	0.2	505.890	1.000			0.0	0.2	490.953	0.988
			0.5	499.297	1.000				0.5	488.867	0.992
			0.0	503.867	0.983				0.0	493.313	0.939
15	0.87	0.2	0.2	511.980	0.999	35	0.34	0.2	0.2	498.837	0.955
			0.5	519.953	0.999				0.5	474.450	0.958
			0.0	496.573	0.924				0.0	480.337	0.867
		0.5	0.2	505.690	0.933			0.5	0.2	482.323	0.881
			0.5	505.243	0.934				0.5	485.560	0.937
			0.0	503.477	0.973				0.0	471.340	0.937
		0.0	0.2	501.563	0.999			0.0	0.2	479.737	0.972
			0.5	500.447	0.989				0.5	471.570	0.970
			0.0	503.967	0.948				0.0	475.583	0.935
20	0.77	0.2	0.2	520.217	0.982	40	0.17	0.2	0.2	489.233	0.950
			0.5	488.570	0.993				0.5	473.457	0.956
			0.0	496.637	0.889				0.0	472.547	0.877
		0.5	0.2	516.623	0.888			0.5	0.2	480.477	0.920
			0.5	516.333	0.926				0.5	481.157	0.960

Table 7: Table 1 Experiment on real-world data.

	MovieLens								GSE103322							
	Low Rankness			Size			CR	Time(s)	Low Rankness			Size			CR	Time(s)
RPSP	**0.82**	**0.8**	**0.76**	1.90E+03	2.50E+03	**2.50E+03**	0.1	52.03	**0.7**	**0.5**	**0.47**	**1.10E+05**	**1.00E+05**	**8.90E+04**	**0.08**	300.6
	0.76	**0.73**	**0.72**	1.70E+03	7.00E+02	7.00E+02			**0.44**	**0.43**	**0.43**	**6.00E+04**	**4.90E+04**	**4.00E+04**		
LLORMA	0.58	0.54	0.52	2.50E+03	2.40E+03	**2.50E+03**	0.06	41.96	0.39	0.39	0.38	1.90E+04	1.90E+04	1.90E+04	0.02	637
	0.52	0.52	0.51	**2.50E+03**	2.30E+03	**2.50E+03**			0.24	0.21	0.19	1.90E+04	1.90E+04	1.90E+04		
SSVD	0.59	0.5	0.5	2.50E+03	**2.60E+03**	2.30E+03	0.06	1.37	0.41	0.29	0.25	1.90E+04	1.90E+04	1.90E+04	0.02	49.59
	0.5	0.5	0.49	2.30E+03	**2.50E+03**	**2.50E+03**			0.24	0.2	0.19	1.90E+04	1.90E+04	1.90E+04		
SPCA	0.48	0.48	0.48	2.30E+03	2.30E+03	2.30E+03	0.06	**0.15**	0	0	0	198	198	198	2.00E-04	**3.55**
	0.48	0.48	0.48	**2.50E+03**	**2.50E+03**	**2.50E+03**			0	0	0	198	198	198		
PLAID	NA	NA	NA	NA	NA	NA	NA	NA	0.31	NA	NA	8.10E+03	NA	NA	1.00E-03	182.84
	NA	NA	NA	NA	NA	NA			NA	NA	NA	NA	NA	NA		
CC	0.51	NA	NA	**6.00E+04**	NA	NA	**0.18**	363	0.28	NA	NA	4.10E+05	NA	NA	0.05	951
	NA	NA	NA	NA	NA	NA			NA	NA	NA	NA	NA	NA		