Interpretable Nonnegative Matrix Decompositions

Saara Hyvönen[*]
Fonecta Ltd.
Helsinki, Finland
saara.hyvonen@fonecta.com

Pauli Miettinen
Helsinki Institute for
Information Technology
University of Helsinki
Finland
pamietti@cs.helsinki.fi

Evimaria Terzi[*]
IBM Almaden
San Jose, CA
USA
eterzi@us.ibm.com

ABSTRACT

A matrix decomposition expresses a matrix as a product of at least two factor matrices. Equivalently, it expresses each column of the input matrix as a linear combination of the columns in the first factor matrix. The interpretability of the decompositions is a key issue in data-analysis tasks. We propose two new matrix-decomposition problems: the nonnegative CX and nonnegative CUR problems, that give naturally interpretable factors. They extend the recently-proposed column and column-row based decompositions, and are aimed to be used with nonnegative matrices. Our decompositions represent the input matrix as a nonnegative linear combination of a subset of columns (or columns and rows) from the input matrix.

We present two algorithms to solve these problems and provide an extensive experimental evaluation where we asses the quality of our algorithms' results as well as the intuitiveness of nonnegative CX and CUR decompositions. We show that our algorithms return intuitive answers with smaller reconstruction errors than the previously-proposed methods for column and column-row decompositions.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data Mining*; F.2.1 [**Analysis of Algorithms and Problem Complexity**]: Numerical Algorithms and Problems—*Computations on matrices*

General Terms

Algorithms, Experimentation

Keywords

Matrix decompositions, column-row decompositions, alternating least squares, local search

[*]This work was partially done when the authors were with Helsinki Institute for Information Technology, Helsinki, Finland.

1. INTRODUCTION

Matrix decompositions have proved to be useful tools for many data-analysis tasks. Broadly speaking, a matrix decomposition takes as input an $m \times n$ matrix A and an integer k and outputs an $m \times k$ *component matrix* C and a $k \times n$ *mixing matrix* X such that A can be approximately represented by the product CX. In that way, the decomposition represents the data by using k components. Naturally, the matrices C and X should be such that they can adequately represent the data in terms of the reconstruction error defined, e.g., as the Frobenius distance $\|A - CX\|_F$.

Matrix decompositions allow for storing matrices C and X instead of the original matrix A. This can save space, especially when the input matrix is dense and $k \ll n$.[1]

In many practical applications it is of interest to express the data using components from the actual data set. Assuming that the columns of A represent the data points, this means that we look for a decomposition of the original matrix into matrices C and X such that the error $\|A - CX\|_F$ is small and, at the same time, the columns of matrix C are actual columns of the original matrix. Then, matrix C tells us what the components are and the mixing matrix X tells us how strongly each component is related to each column of the original matrix A. Such a decomposition is known as the CX decomposition. The corresponding decomposition where the matrix A is decomposed into three matrices, C, U and R, where C contains a subset of the columns, R a subset of the rows, and U is a mixing matrix, is called the CUR decomposition.

The CX and CUR decompositions have received increasing attention in the data-analysis community [2, 5, 6]. One of their advantages is that they are very space efficient since they preserve data sparsity: if the input matrix is sparse then so are C and R. At the same time, CX and CUR decompositions give a relatively intuitive interpretation of the original data. Consider matrix A where the columns correspond to newsgroup categories and rows to words used in newsgroups' posts; each entry (i, j) of matrix A gives the frequency of word i in newsgroup j. Further assume that the columns that correspond to *news*, *religion*, and *sports* categories are selected to be in C. Then, the CX decomposition can express any other column, e.g., a column that corresponds to newsgroup about *golf*, as being 0.9 of *sports*, -0.6 of *religion*, and 0.3 of *news*. Knowing the meaning of the components in matrix C (e.g., sports, news or religion) increases the interpretability of the decomposition.

[1]This is an assumption that we will follow throughout the paper.

As a real-world example, consider the **dialect** dataset, discussed in Section 4.3. The rows of the data represent the dialect features and the columns represent the municipalities of Finland so that element (i, j) tells whether feature i is used in municipality j or not. When doing a `CUR` decomposition to this data the columns of C represent the prototypical municipality of a certain dialect group, while the rows of R represent the prototypical dialect features of each dialect group. In this paper, we take the `CX` (and `CUR`) decompositions one step further by adding a nonnegativity constraint on the mixing matrices: matrices X (in `CX`) and U (in `CUR`) are required to be *nonnegative*.

In many data-analysis tasks the data matrices have only nonnegative entries; such entries correspond, for example, to counts of event occurrences. In cases where the input matrix is nonnegative, the additional nonnegativity requirement is not only reasonable, but it also helps in the interpretability of the results. For example in the aforementioned newsgroup data expressing the column about *golf* a being 0.8 of *sports*, 0 of *religion*, and 0.1 of *news* seems more intuitive. For the **dialect** data the nonnegative mixing matrices naturally express the strength of dialect features in municipalities showing, e.g., the smooth transition from a dialect to other.

In this paper we define the nonnegative `CX` and `CUR` decompositions (`NNCX` and `NNCUR` respectively) and give two algorithms to find them. Via extensive experimental evaluation we show that the algorithms work well in practice: they give interpretable decompositions and achieve low reconstruction errors. In fact, with nonnegative data our algorithms achieve lower reconstruction errors than general `CX` and `CUR` decomposition methods. In some cases this is true even when the latter are not forced to satisfy our nonnegativity constraint.

The rest of the paper is organized as follows. Problem definitions and some facts regarding the computational complexity of the `NNCX` and `NNCUR` problems are given in Section 2. In Section 3 we present our algorithms. Section 4 contains the results of our experiments and Section 5 gives a review of the related work in the area. We conclude the paper in Section 6.

2. THE PROBLEMS

In this section we provide some background on the `CX` and `CUR` decompositions and define their nonnegative variants, which are the main focus of this paper. We also outline some basic properties of these variants and study their computational requirements.

2.1 Problem definitions

Let A be an $m \times n$ matrix. We denote the i-th column of A by A^i. Similarly for $I \subseteq \{1, \ldots, n\}$ we use A^I to refer to the submatrix of A that contains only the columns of A indexed by the indices in I. We use $\|\cdot\|_F$ to denote the Frobenius norm of a matrix, $\|A\|_F = \sqrt{\sum_i \sum_j a_{ij}^2}$, and $\mathbb{R}_+$ to denote the nonnegative real numbers.

For an integer k, we define the *column-based low-rank approximation* of A to be the matrix $A' = CX$, where C is an $m \times k$ matrix whose columns consist of k columns of the input matrix A, and mixing matrix X is in $\mathbb{R}^{k \times n}$. Matrices C and X define a `CX` decomposition of A. The cost of such a decomposition is defined to be

$$CX(A, k, C, X) = \|A - CX\|_F.$$

The `CX` problem, given k, asks for the matrices C and X such that $CX(A, k, C, X)$ is minimized. We use $CX^*(A, k)$ to denote this minimum value. The complexity of finding the optimal solution to the `CX` problem is unknown [5]. However, a recent result [3] strengthens the intuition of the NP-hardness of the problem.

The `CUR` decomposition is an extension of the `CX` decomposition. For an input matrix A and integers k and r, we define the *column-row based low-rank approximation* of A to be the matrix $A' = CUR$ where C is an $m \times k$ matrix with k columns from A, R is an $r \times n$ matrix with r rows from A, and $U \in \mathbb{R}^{k \times r}$ is a mixing matrix. Matrices C, R, and U define a `CUR` decomposition of A. The cost of such a decomposition is defined to be

$$CUR(A, k, r, C, U, R) = \|A - CUR\|_F.$$

As in the `CX` problem, given k and r, the `CUR` problem asks for matrices C, R, and U such that $CUR(A, k, r, C, U, R)$ is minimized. We denote by $CUR^*(A, k, r)$ this minimum value. Solving the `CUR` problem is at least as hard as solving the `CX` problem.

Adding the nonnegativity constraints in the `CX` and `CUR` decompositions leads to the definitions of the *nonnegative CX* (`NNCX`) and *nonnegative CUR* (`NNCUR`) problems, which are the focus of this paper.

Problem 2.1. *Given a matrix $A \in \mathbb{R}_+^{m \times n}$ and an integer k, find an $m \times k$ matrix C of k columns of A and a matrix $X \in \mathbb{R}_+^{k \times n}$ minimizing*

$$CX_+(A, k, C, X) = \|A - CX\|_F.$$

We refer to Problem 2.1 as the `NNCX` problem and to the `CX` decomposition where $x_{ij} \in \mathbb{R}_+$ as the `NNCX` decomposition of A. We also use $CX_+^*(A, k)$ to denote the minimum value of the $CX_+(A, k, C, X)$ function. Notice that the matrix C can be interpreted as a set of points generating a convex cone and the goal is to minimize the sum of distances from other columns of A to this cone.

The definition of the `NNCUR` problem is similar to that of `NNCX`.

Problem 2.2. *Given a matrix $A \in \mathbb{R}_+^{m \times n}$ and integers k and r, find an $m \times k$ matrix C of k columns of A, an $r \times n$ matrix R of r rows of A, and a matrix $U \in \mathbb{R}_+^{k \times r}$ minimizing*

$$CUR_+(A, k, r, C, U, R) = \|A - CUR\|_F.$$

A solution to the `NNCUR` problem is an `NNCUR` decomposition of the input matrix A. We denote the cost of the optimal `NNCUR` decomposition by $CUR_+^*(A, k, r)$.

From the above definitions, it is easy to derive a set of propositions that summarize the relationships between the different decompositions.

Proposition 1. *For a nonnegative matrix A and integers k and r, it holds that*

$$CX^*(A, k) \leq CX_+^*(A, k) \qquad \text{and}$$
$$CUR^*(A, k, r) \leq CUR_+^*(A, k, r).$$

Proposition 2. *For a nonnegative matrix A and integers k, k' such that $k \leq k'$ and r, r' such that $r \leq r'$, it holds that*

$$CX_+^*(A, k) \geq CX_+^*(A, k') \qquad \text{and}$$
$$CUR_+^*(A, k, r) \geq CUR_+^*(A, k', r').$$

A trivial upper bound for $CX_+^*(A,k)$ is obtained by selecting C to consist of the k columns of A with the largest norms and assigning $x_{ij}=1$ if $C^i = A^j$ and zero otherwise. This yields the following bound.

PROPOSITION 3. *Let J be the indices of the $n-k$ columns of A with the smallest norms. Then $CX_+^*(A,k) \le \|A^J\|_F$.*

2.2 Problem complexity

As mentioned above, the complexity of the CX and CUR problems is unknown. However, we can show that given the component matrices, the mixing matrices X and U can be computed in polynomial time. We present our results in terms of the CX and NNCX decompositions, but all results apply *mutatis mutandis* to CUR and NNCUR.

Notice first that if the matrix C is given, then finding the matrix X that minimizes $CX(A,k,C,X)$ can be done in polynomial time by setting $X = C^\dagger$, where $C^\dagger$ is the Moore–Penrose pseudoinverse[2] of C [11]. An analogous result for nonnegative X is given in the next proposition.

PROPOSITION 4. *Given a matrix $A \in \mathbb{R}_+^{m\times n}$, an integer k, and an $m\times k$ matrix C of k columns of A, the matrix $X \in \mathbb{R}_+^{k\times n}$ minimizing $CX_+(A,k,C,X)$ can be computed in polynomial time.*

For the proof notice that finding the NNCX decomposition of a matrix translates into a convex quadratic programming problem with linear constraints. This can be solved in polynomial time [12].

Therefore, for a given matrix C, finding the optimal X can be done in polynomial time for both CX and NNCX decompositions. Similarly, given C and R, finding the optimal matrix U for the CUR and NNCUR decompositions can also be done in polynomial time. These observations give a hint that the difficulty of Problem 2.1 (or Problem 2.2) lies in finding the correct C (or C and R).

3. ALGORITHMS

In this section we give two approximation algorithms for the NNCX and NNCUR problems. We mainly focus our attention on the algorithms for the NNCX problem. We then show how we can solve the NNCUR problem using the algorithms for the NNCX problem as black boxes.

3.1 Algorithms for the NNCX problem

To solve the NNCX problem, one could take any existing algorithm for the CX problem and then apply the nonnegativity constraint to matrix X (for existing algorithms, see, e.g., [2, 6]). Instead of taking this approach we propose two new algorithms for the NNCX problem: LOCAL and ALS.

The LOCAL algorithm is a local-search heuristic that takes as input a nonnegative matrix A and an integer k and computes matrices C and X. The algorithm initially starts with a random subset of columns of A forming the matrix C. It then repeatedly replaces a column from C with another column from A, given that such a swap decreases the cost of the NNCX decomposition. The algorithm stops when no more improvements can be made, or when a limit on the number of iterations is reached. The user can select the maximum number of iterations based on her demands, use some predefined default value, or omit the whole stopping criterion. Algorithm 1 sketches the main steps of the LOCAL algorithm.

Algorithm 1 The LOCAL algorithm for NNCX.

```
   Input: An $m \times n$ nonnegative matrix $A$ and an integer $k$.
   Output: Matrices $C$ and $X$ that approximate $A$.
1: $J = k$ random distinct column indices in $\{1,\dots,n\}$
2: $C = A^J$;
3: $[X, \mathrm{err}] =$ Solve_X_Given_C$(A, C)$
4: while err decreases AND max iterations are not reached do
5:     for $j \in J$ do
6:         $j' = \arg\min_{c\in\{1,\dots,n\}\setminus J}\{\mathrm{err} \mid [X,\mathrm{err}] =$ Solve_X_Given_C$(A, A^{(J\setminus\{j\})\cup\{c\}})\}$
7:         $J = (J \setminus \{j\}) \cup \{j'\}$
8:         $C = A^J$;   $X =$ Solve_X_Given_C$(A, C)$
```

Algorithm 2 The ALS algorithm for NNCX.

```
   Input: An $m \times n$ nonnegative matrix $A$ and an integer $k$.
   Output: Matrices $C$ and $X$ that approximate $A$.
1: $J = k$ random column indices from $\{1,\dots,n\}$
2: $\tilde{C} = A^J$;
3: while err decreases AND max iterations are not reached do
4:     $[X, \mathrm{err}] =$ Solve_X_Given_C$(A, \tilde{C})$
5:     $[\tilde{C}, \mathrm{err}] =$ Solve_C̃_Given_X$(A, X)$
6: $J =$ match_columns$(\tilde{C}, A)$
7: $C = A^J$
8: $[X, \mathrm{err}] =$ Solve_X_Given_C$(A, C)$
```

At every step we fix a set of columns that form the matrix C, and we need to evaluate the cost of the solution and the nonnegative matrix X induced by this particular C. This task is handled by the function Solve_X_Given_C. Given nonnegative matrices A and C, Solve_X_Given_C returns a nonnegative matrix X such that CX is close to A. See below for more discussion about this function and its counterpart Solve_C̃_Given_X.

Another important point is that the LOCAL algorithm is not bounded to the NNCX problem, but it can also be used as an alternative solution to the standard CX problem simply by not requiring X to be nonnegative. As we will see in Section 4, using LOCAL for the standard CX problem gives in many settings better results than standard algorithms used for the CX problem.

The second algorithm we propose for the NNCX problem comes with the name ALS and is inspired by the alternating least squares method used in the NMF algorithms [1]. A sketch of the algorithm is given in Algorithm 2.

The idea is that we first pick a random set of k columns from the original matrix to construct a matrix $\tilde{C}$. Then, we use this $\tilde{C}$ to compute the nonnegative matrix X using the routine Solve_X_Given_C. In turn, we compute a new nonnegative matrix $\tilde{C}$ using the current X in an analogous way of computing X given $\tilde{C}$. This new $\tilde{C}$ does not have to contain columns of A (hence $\tilde{C}$, not C). This computation is handled by the routine Solve_C̃_Given_X.

The loop terminates either when the alternating process gives no more improvement in the error function, or when a certain number of iterations has been reached. Alternatively, the stopping criteria can be changed so that the **while** loop terminates when the improvement in the error goes below some threshold. Again, the user can supply the algorithm with the stopping criteria best suited with her needs or rely on default values.

[2]Sometimes C^+ is used to denote the pseudoinverse, but we restrict the use of $+$ to denote nonnegativity.

After the **while** loop, matrix $\tilde{C}$ might not – and in general does not – contain columns of A. Thus, we need an additional post-processing step on line 6. The task is to find k distinct columns of A that are as close as possible to the columns of $\tilde{C}$. This task is handled by the routine `match_columns` on line 6, and it involves finding a solution to a standard maximum matching problem on bipartite graphs. This problem can be solved optimally in polynomial time using the Hungarian method [13]. Greedy alternatives also work well in practice.

3.1.1 Solving X and $\tilde{C}$

The most straightforward way of implementing methods `Solve_X_Given_C` and `Solve_C̃_Given_X` without the nonnegativity constraint is to use the Moore–Penrose pseudoinverse. By Proposition 4, polynomial-time implementations of the algorithms with nonnegativity constraints are possible using convex quadratic programming (CQP). However, the computational requirements of CQP routines would make the overall approach highly unscalable, as we need to use it in the inner loops of both algorithms. As a post-processing step, i.e., fine-tuning X after C is fixed, CQP could be used in some cases. For the sake of a fair comparison, we have not used CQP in our experiments, except in Section 4.2.1.

In order to keep the algorithms scalable for large datasets, even with some cost in the quality of the results, we use the following method: In `Solve_X_Given_C`(A, C), we first compute $X = C^\dagger A$, where $C^\dagger$ is the Moore–Penrose pseudoinverse of C. Then we set all negative values of X to 0. Equivalently, when solving $\tilde{C}$ given X, we compute $\tilde{C} = AX^\dagger$ and set the negative values of $\tilde{C}$ to 0. Our *projection-based* approach is in fact a common practice employed in many `NMF` algorithms [1]. In Section 4 we will show that the effect the projection step has to the quality of the results can be very small or even nonexistent.

Our projection approach is very simple, and it is possible to improve it with some simple ideas. For example, one could project all values of X that are smaller than some $\varepsilon \geq 0$ to 0. We have noticed that a careful selection of ε can indeed yield smaller errors. However, in order to avoid extra parameters we have not utilized this trick in our experiments.

3.1.2 Convergence of the algorithms

The Local algorithm will eventually converge to some stationary point since there is only a limited, albeit exponential, number of different matrices C. However, this stationary point is not guaranteed to be a local optimum. The convergence of ALS is a more complicated issue. Had we not the requirement for the columns to be from the original matrix, the results reported in [1] would directly apply here. That is, the **while** loop in line 3 of the ALS algorithm converges to local optima given that the optimal solution is computed in every step. However, the last step of mapping the columns of $\tilde{C}$ to the closest columns of the original matrix A (line 6), prevents us from claiming anything about the optimality of the obtained solution.

3.1.3 Time complexity

The time complexity of our algorithms is dictated by the time complexity of `Solve_X_Given_C` and `Solve_C̃_Given_X`, both requiring the computation of Moore–Penrose pseudoinverse[3] for an $m \times k$ (or $k \times n$) matrix, and a multiplication of two matrices of size $m \times k$ and $k \times n$. Using `SVD` for pseudoinverse, this is doable in time $O(nk^2)$ (assuming $n = m$) [11]. That is also the time needed for a single iteration of the ALS algorithm. A single iteration of Local takes time $O((n-k)nk^3)$. Assuming that the number of iterations is constant, these are also the final time complexities, albeit with possibly large hidden constants.

[3]Here we are using the projection method.

3.2 Algorithms for the NNCUR problem

For the `NNCUR` problem we use the algorithms proposed in the previous section as subroutines. Suppose $\mathcal{A}$ is an approximation algorithm for the `NNCX` problem. We use $\mathcal{A}$ to compute an approximate solution to the `NNCUR` problem. First, we solve the `NNCX` problem for the input matrix A and for its transpose, A^T. As a result, we have matrices C and R containing a subset of columns and a subset of rows of A, respectively. Given C and R, we compute the matrix U. Again, instead of using the convex optimization, we use the pseudoinverse so that U is the matrix $C^\dagger A R^\dagger$ with all negative values set to 0.

The above procedure is not specific to the `NNCUR` problem, but it can be used for finding approximate solutions to the general `CUR` problem as well.

4. EXPERIMENTAL EVALUATION

In this section we give an extensive experimental study of the performance of our algorithms. For this, we test our algorithms on both synthetically-generated and real datasets. The main goals of our experiments are twofold: (1) to show that `NNCX` and `NNCUR` decompositions, in general, are viable decomposition schemes; and (2) to show that our algorithms, in particular, achieve good and intuitive results. The synthetic data is used to asses the quality of our algorithms' results, and with it we study the effects of projection versus convex optimization; the effects of basis size; and the effects of noise. The real data serves both purposes of the experiments: with it we study how well our algorithms perform against previously-proposed methods for `CX` and `CUR` decompositions; what are the effects of the nonnegativity constraint and how meaningful it is; and are the results intuitive.

Our evaluation metric is the reconstruction error, i.e., the Frobenius norm of the difference between the input matrix A and the matrix A' constructed by the decomposition, $\|A - A'\|_F$ with $A' = CX$ or $A' = CUR$.

4.1 Other algorithms

Since this is the first study to address the `NNCX` and `NNCUR` problems, we compare our algorithms against two previously-proposed algorithms for `CX` and `CUR` decompositions. These algorithms were designed without the nonnegativity constraint in mind. The first of these two algorithms is called 844 and is due to Berry et al. [2]. It is a greedy algorithm that selects the k-th column in C based on the $k-1$ already selected columns. The other algorithm, DMM, is due to Drineas et al. [7], and we use the Exactly version of it. The DMM algorithm contains two algorithms: one for the `CX` problem and one for `CUR`. Both algorithms are, however, based on the same idea of randomly sampling the columns in C (and rows in R) with respect to probabilities defined by the singular vectors of the input matrix. Drineas et al. [5, 6, 7] were able to prove that their algorithms can, with high probability, achieve a reconstruction error that is at most

$(1+\varepsilon)$ times the optimal reconstruction error achieved by SVD. Alas, to achieve this result against SVD's rank-k approximation, their algorithms need k' columns. Parameter k' depends on k, ε, and δ, the probability of success, but it is usually orders of magnitude larger than k. In our cases however, we are interested in the case where the number of columns is fixed and relatively small.

The time complexity of DMM is the same as computing a rank-k approximation with SVD [7]. 844's time complexity is not clearly stated in [2], but it is empirically shown to be somewhat faster than SVD.

For a fair comparison, we modify 844 and DMM with the additional step of restricting matrices X and U to be nonnegative. We refer to these methods as the nonnegative 844 and nonnegative DMM, respectively. However, we also make experiments without the nonnegativity constraint in 844 and DMM. We also compare our methods against a straightforward heuristic based on the k-means clustering algorithm that we call KMEANS. KMEANS first performs k-means clustering of the columns of A. Then it constructs matrix C by selecting the columns of A that are the closest to the centroids of the clustering. Given C it then solves X.

We also report the reconstruction errors of SVD and NMF decompositions as baselines. The algorithms used to find SVD and NMF decompositions are denoted by SVD and NMF, respectively. Note that any feasible solution to NNCX is also a feasible solution to NMF, and also an at-most-rank-k approximation of the input matrix. Thus, the optimal NMF and rank-k approximations are always at least as good as the optimal NNCX approximation, and can be much better. Therefore the reconstruction error of SVD is a lower-bound for the error of the optimal NNCX (or NNCUR) decomposition. In fact, the reconstruction error of SVD should be expected to be far smaller than the reconstruction error of the optimal NNCX (or NNCUR) decompositions. Our experiments demonstrate that in most of the cases our algorithms achieve errors very close to the error of SVD which means that not only our algorithms give high-quality results, but also that NNCX and NNCUR decompositions are reasonable decompositions in practice.

Although the optimal solution to the nonnegative matrix factorization (NMF) problem is in principle a lower bound to the optimal solution of NNCX (NNCUR) problem, the algorithm for NMF we use here (the original multiplicative algorithm [1], NMF) is not optimal, and as a such, does not present any lower bounds for our algorithms' performance. In fact, in some cases our algorithms are better than the NMF algorithm. This illustrates that at least in those cases our algorithms for NNCX and NNCUR are closer to their optimum than the corresponding NMF algorithm.

In all experiments the maximum number of iterations was 300 for LOCAL and 200 for ALS and NMF. However, neither LOCAL nor ALS ever met their maximum, as both converged in at most dozens of iterations. All algorithms were implemented as Matlab functions. For 844 we used the implementation provided with the article [2]. The other algorithms are our implementations, save SVD for which we used the Matlab's built-in version.

4.2 Experiments on synthetic datasets

We generated the synthetic datasets as follows. Consider the task of generating a data matrix A of size $m \times n$ with k columns as its basis. First we generated the columns $A^1, \dots, A^k$ as i.i.d. nonnegative random vectors. The rest of the columns were generated as random linear combinations of the first k columns. The size of the matrices is 200×150. In order to test the performance of the algorithms we added a noise matrix N^ℓ: $(N^\ell)_{ij}$ is nonzero with probability $\ell \in (0,1)$, an the nonzero entries are uniformly distributed in the unit interval.

The method for generating synthetic datasets for testing the NNCUR methods is a bit more complex. In fact, we restrict our data-generation process to the creation of matrices A with a special CUR structure in which U is a square identity matrix of size $k \times k$.[4] For input $k = r$ the data-generation process outputs a matrix A of size $m \times n$ such that $A = CR$, where $C = \begin{pmatrix} I_{k\times k} \\ B_{(m-k)\times k} \end{pmatrix}$ and $R = \begin{pmatrix} I_{k\times k} \mid B'_{k\times(n-k)} \end{pmatrix}$. The matrices B and B' have random entries uniformly distributed in the unit interval. We also added noise matrices N^ℓ, similar to those used with NNCX.

We used two variables in the data-generation process that allowed us to better study the behavior of the algorithms: the noise level ℓ and the size of the decomposition, k. First, we fixed $\ell = 0.05$ and let $k = 2, \dots, 20$. Then, we fixed $k = 10$ and let $\ell = 0.01, \dots, 0.5$. In all cases k refers to the actual value of k used to create the data. In all experiments, including the ones with real datasets, ALS, LOCAL, NMF, and DMM were executed 3 times with each dataset and the best result was selected. This was done to avoid the adverse effect of a bad random starting point. In Figures 1–3 every point represents an average over 5 datasets generated with identical parameters.

4.2.1 *Convex quadratic programming vs. projection*

We start with an experiment that aims at justifying our choice to use the projection approach instead of convex quadratic optimization when solving X and $\tilde{C}$. For the experiment we fixed $k = 10$, and let the noise level vary between 0.01 and 0.5. We used the `cvx` software[5] for the convex optimization.[6]

Figure 1 shows the reconstruction errors of SVD, ALS, LOCAL, ALS CVX, LOCAL CVX and OPT CVX algorithms. The ALS and LOCAL are the versions of our algorithms in which the projection approach was used. ALS CVX and LOCAL CVX are the versions of our algorithms in which we used convex optimization in the routines `Solve_X_Given_C` and `Solve_C̃_Given_X`. Finally, since these are experiments on synthetic data we also report OPT CVX that uses the correct columns (used in the data-generation process) to form matrix C and computes X using convex optimization.

From Figure 1 we observe that the ALS algorithm constantly achieves the optimal solution even with the projection. With low levels of noise, LOCAL is also reporting optimal results, but as the noise level increases, so does the LOCAL's error. As the error increases, convex optimization benefits the algorithm more. Figure 1 also shows that while the error curve of ALS follows that of SVD the latter is still clearly lower. Overall, the experiment verified our expectation that using projection instead of convex optimization

[4]We are not aware of any technique that could generate data that can have an arbitrary NNCUR (or even CUR) decomposition with zero error.

[5]`http://www.stanford.edu/~boyd/cvx/`

[6]Convex quadratic optimization is a special case of convex optimization.

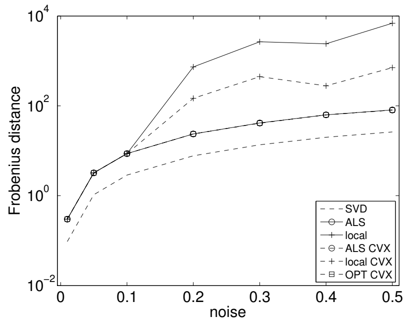


Figure 1: Reconstruction error in the `NNCX` decomposition as a function of noise. OPT CVX, ALS CVX and Local CVX use convex optimization as a post-processing step to compute X, and OPT CVX knows the correct columns to put in C. ALS and Local (with and without CVX) were restarted 3 times for each dataset, and the best result was selected. All points represent an average over 5 datasets with identical parameters. Logarithmic scale in y-axis. Lines for ALS, ALS CVX, and OPT CVX are on top of each other.

does not deteriorate the quality of the results. Therefore, we adopted it for the rest of our experiments.

4.2.2 *Effects of noise and basis size*

Figure 2 shows the reconstruction errors of the `NNCX` decompositions with synthetic datasets as a function of the noise level ℓ and the value of k used in the data-generation process (Figures 2(a) and 2(b)). ALS and Local perform consistently far better than Kmeans and the nonnegative versions of 844 and DMM. Note that in all of the experiments at least one of our algorithms achieves a reconstruction error very close to that of SVD. Finally, it must be noted that the relatively poor performance of DMM does not violate the results in [5, 6, 7] – i.e., that DMM should be at most twice as bad as SVD – as for that claim to hold, one must have $k' \gg k$ columns in C instead of k. Indeed, all algorithms return the same number of columns in C.

In Figure 2(a), as k increases, the quality of ALS decreases while the quality of Local is almost unchanged, but in Figure 2(b), when noise level increases, the behavior of the algorithms is reversed. One must note that the value of k represents both the parameter for the algorithms, and the number of i.i.d. vectors used to create the data. Thus the behavior of the ALS algorithm in Figure 2(a) does not contradict Proposition 2.

Figure 3 shows the reconstruction errors for `NNCUR`. Our algorithms are again consistently better than all of the other `NNCUR` methods. Moreover, their reconstruction errors are again very close to that of SVD and NMF. ALS is clearly the best method with Local being the second (though considerably worse).

4.3 Real data

To compare the performance of the different methods to solve the `CX` and `CUR` problems on real data we have used four different datasets. The **newsgroup** dataset is a subset of the 20Newsgroups collection[7] containing frequencies of 800 words in 400 messages from 4 newsgroups ('sci.crypt', 'sci.med', 'sci.space', and 'soc.religion.christian'). The data was preprocessed with the Bow Toolkit[8], by Andrew McCallum, with stemming and stop word removal, and taking 800 most frequent words.

The **dblp** dataset contains conference–author pairs giving the number of articles authors have published in conferences. The data is collected from the DBLP Bibliography database[9]. For the purposes of our experiments, we selected 19 conferences, viz. WWW, SIGMOD, VLDB, ICDE, KDD, SDM, PKDD, ICDM, EDBT, PODS, SODA, FOCS, STOC, STACS, ICML, ECML, COLT, UAI, and ICDT. We then extracted all authors that had in total at least 2 publications in the selected conferences, yielding a 19×6980 matrix.

The **dialect** dataset [8, 9] consists of the distribution of 1334 dialect features (rows) across 506 municipalities of Finland (columns). Each municipality is additionally characterized by two spatial coordinates.

The **Jester joke** dataset [10] contains users' ratings of 100 jokes in a real range $[-10, 10]$. We considered only users that had rated all 100 jokes, yielding $14\,116 \times 100$ matrix. We made this matrix nonnegative via adding 10 to all ratings so that the ratings are in the range $[0, 20]$. Apart from this nonnegativization step, the same dataset was used in [7]. This dataset was used to mimic an experiment from [7].

4.3.1 *Reconstruction errors*

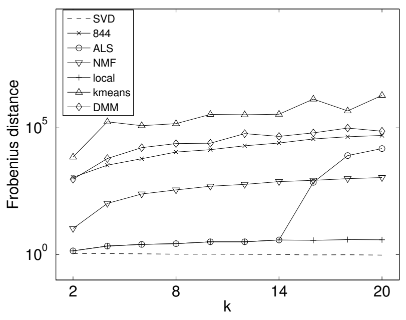
We present the reconstruction errors for the **newsgroup** dataset in Figure 4. Figure 4(a) shows the errors for `CX` and `NNCX` decompositions and Figure 4(b) for `CUR` and `NNCUR` decompositions. From Figure 4(a) we observe that the Kmeans and DMM methods perform consistently worse than the other methods and that Local and ALS are at least as good as 844. These results verify that the solid performance of our algorithms in terms of reconstruction error is not an artifact of the synthetic data, but it is also observed in real-life datasets. Note also, that the smallest error achieved for the `NNCX` decompositions is very close to that of NMF and SVD. In other words, restricting ourselves to `NNCX` decompositions does not increase the reconstruction error much compared to arbitrary (nonnegative) decompositions. It seems that the **newsgroup** dataset indeed has an inherent `NNCX` structure.

The observation that `NNCX` and `NNCUR` decompositions are natural in real datasets is strengthened by the experimental results of Figure 4(b). This figure shows the reconstruction errors achieved by the Local, ALS and 844 algorithms for both `CUR` (algorithms ALS, Local, 844) and `NNCUR` decompositions (algorithms ALS_nn, Local_nn and 844_nn). The reconstruction errors of `CUR` and `NNCUR` are very close to each other, implying that not only is the `CUR` decomposition natural, but also the nonnegativity constraint is natural in this setting. Moreover, the `NNCUR` decompositions obtained by Local and ALS are better than the `CUR` decompositions (that is, without the nonnegativity constraint) obtained by 844. This shows that when the data admits to `NNCUR` decomposition, our methods can outperform the previously-proposed methods even if the latter are not restricted to nonnegative decompositions. We believe that this further emphasizes the fact that our algorithms are producing good results and that studying `NNCX` and `NNCUR` decompositions is meaningful.

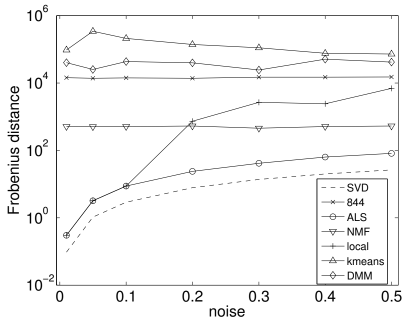
[7]`http://people.csail.mit.edu/jrennie/20Newsgroups/`

[8]`http://www.cs.cmu.edu/~mccallum/bow/`

[9]`http://www.informatik.uni-trier.de/~ley/db/`

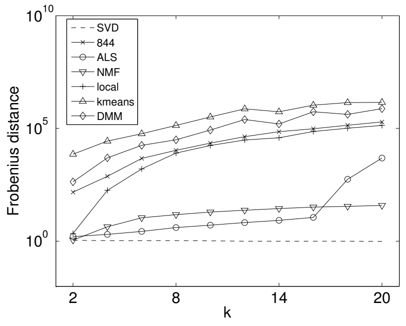


(a)

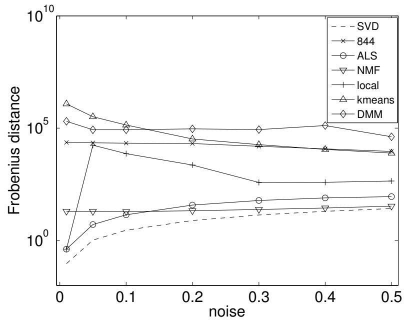


(b)

Figure 2: Reconstruction errors in the CX and NNCX decompositions as a function of (a) k and (b) noise. Best results from 3 restarts is selected and the points represent an average over 5 datasets with identical parameters. Logarithmic scale in y-axes.

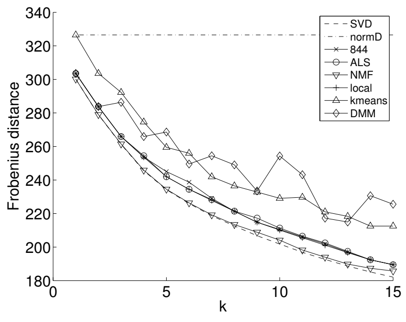


(a)

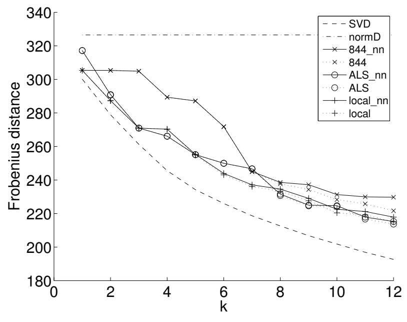


(b)

Figure 3: Reconstruction errors in CUR and NNCUR decompositions as a function of (a) k and (b) noise. Best results from 3 restarts is selected and the points represent an average over 5 datasets with identical parameters. Logarithmic scale in y-axes.

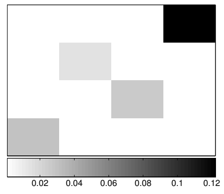


(a)

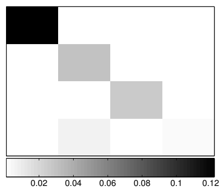


(b)

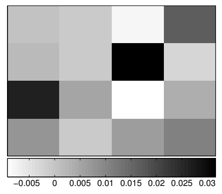
Figure 4: Errors for the NNCX (a), and CUR and NNCUR (b) decompositions of **newsgroup** data. In (b) solid lines represent NNCUR decompositions. Best of 3 restarts is reported.



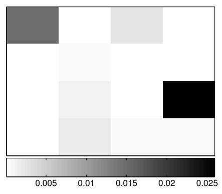
(a) ALS with NNCUR decomposition



(b) LOCAL with NNCUR decomposition



(c) DMM with CUR decomposition



(d) 844 with CUR decomposition

Figure 5: The **newsgroup** data; mixing matrices U of NNCUR decompositions for algorithms ALS and LOCAL and CUR decompositions for algorithms DMM and 844. Column and row labels for the matrix elements are in Tables 1 and 2, respectively.

Table 1: Terms selected by algorithms in `CUR` decompositions of **newsgroup** data with $k = 4$. Ordering corresponds to the order of matrix columns in Figure 5.

ALS	Local	DMM	844
diet	atlant	year	venu
god	diet	launch	god
encrypt	encrypt	studi	launch
atlant	god	system	encrypt

Table 2: Newsgroups corresponding to the news selected by algorithms in `CUR` decompositions of **newsgroup** data with $k = 4$. Ordering corresponds to the order of matrix rows in Figure 5.

ALS	Local	DMM	844
space	space	crypt	space
christian	med	med	med
crypt	crypt	space	crypt
med	med	crypt	med

4.3.2 Quality and interpretability of results

A key motivation of this work was to find factors and coefficients which have a clear interpretation. That is, all matrices C, U, and R (or C and X) must be intuitive. To study this, we start with the results from `CUR` (for DMM and 844) and `NNCUR` decompositions (for ALS and Local) of the **newsgroup** dataset. Table 1 gives the terms selected by algorithms with $k = 4$ (a term is selected if the row corresponding to it appears in the matrix R), and Table 2 gives the newsgroups related to the news selected by algorithms. Finally, Figure 5 shows the mixing matrices U obtained from these four algorithms. The columns of Tables 1 and 2 give names to the columns and rows of matrices in Figure 5.

From Table 1 one can easily see that ALS and Local select one term corresponding to each newsgroup. This one-to-one correspondence between terms and newsgroups is further confirmed by examining the mixing matrices. Especially for ALS the one-to-one correspondence between the rows (news articles) and columns (terms) is clearly observable in matrix U (Figure 5(a)), the rows of which represent from top to bottom the newsgroups space, christian, crypt and med. So in this case U is simply a weighted permutation matrix. For Local (Figure 5(b)) this is almost the case, but not quite. A reason for this is visible from Table 2: Local has failed to select a news article from the christian newsgroup, so the weight of the element corresponding to the term picked from this newsgroup is small. A similar phenomenon is found in 844's column in Table 2 and, indeed, in Figure 5(d). The mixing matrix for DMM (Figure 5(c)) is clearly different from the rest: all selected terms are used for all groups with some nonzero weight. Note, that DMM also uses negative values in U, e.g., by relating newsgroup 'sci.space' negatively to term 'studi'.

We used only `NNCX` decomposition with the **dblp** dataset since it only has 19 rows, and we only report the columns selected by ALS and Local with $k = 6$ (Table 3). The columns are again very intuitive; for example, in ALS's results Umesh V. Vazirani is selected to represent theoretical computer scientists that publish in focs and stoc. In Local's results, Vazirani's role is given to Leonid A. Levin. Equivalently, John Shawe-Taylor (in Local's results) represents a group of machine learners, and Hector Garcia-Molina (in ALS) and Divesh Srivastava (in Local) a group of data-management researchers.

Table 3: Results for **dblp** data using ALS and Local with $k = 6$.

ALS	Local
Naoki Abe	Divesh Srivastava
Craig Boutilier	Leonid A. Levin
Uli Wagner	Souhila Kaci
Umesh V. Vazirani	Xintao Wu
Hector Garcia-Molina	Uli Wagner
Dirk Van Gucht	John Shawe-Taylor

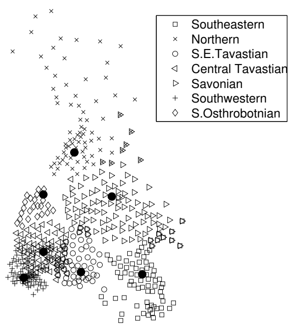


Figure 6: The **dialect** data. The symbols show the spread of each feature selected in the rows in `NNCUR`. The solid dots mark the columns selected by `NNCUR`: each of those correspond to one municipality. The ALS algorithm was used.

We further demonstrate the usefulness of our decompositions using the **dialect** dataset. The `NNCUR` decomposition of this dataset reflects the underlying dialect groups. In Figure 6 we plot with different (non-solid) symbols the features (rows) picked in the `NNCUR`. Each symbol appears in the map coordinates of a municipality if the corresponding dialect feature has a non-zero value in this municipality. The municipalities selected are plotted as solid dots. There is little overlap between the spreads of the selected features, meaning that the algorithm picks the rows that best characterize certain regions.

4.3.3 The Jester joke dataset

The purpose of our final experiment on the **Jester joke** dataset was to mimic the experimental setting of the recent paper by Drineas et al. [7]. Thus, the setup of this experiment differs somewhat from our other experiments; here we study the reconstruction accuracy of `CX` and `NNCX` decompositions with different values of k relative to that of SVD with a *fixed* value of k. Namely, we let k vary from 5 to 30 for ALS, Local, and DMM, and report the Frobenius error of this decomposition divided by the reconstruction error of `SVD` with $k = 5$. All experiments were repeated 3 times and the best result is reported. Thus our results are analogous to those in [7, Figure 3] with the exception of the 'nonnegativization' of the matrix. It should be noted that, as the decompositions can have rank higher than 5, they can achieve reconstruction errors below that of `SVD`. The results are given in Figure 7.

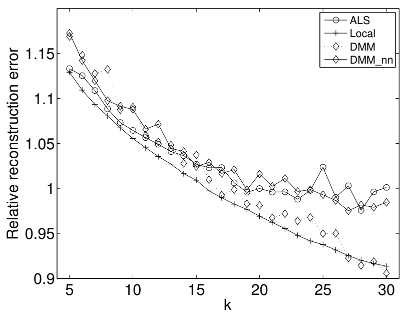


Figure 7: Reconstruction error of CX and NNCX decompositions with **Jester joke** data relative to that of SVD with $k = 5$. Solid lines represent NNCX decompositions. All experiments were repeated 3 times and the best result is reported.

Figure 7 shows that in this experiment Local is the best method. Only DMM without nonnegativity constraint in X can achieve comparable results. When the matrix X in the decomposition returned by DMM is projected to the nonnegative quadrant, the results of DMM do not have much improvements after $k = 20$. This behavior of DMM resembles that of ALS. This experiment again confirms that with nonnegative data our algorithms can perform better than general CX decomposition methods.

5. RELATED WORK

To the best of our knowledge we are the first to introduce and study algorithms for the nonnegative versions of CX and CUR decompositions. However, matrix decompositions in general have been a recurrent research theme.

One of the best-known matrix-decomposition approaches is the singular value decomposition (SVD) [11]. SVD has been utilized in a wide range of data-mining applications [4]. SVD gives the optimal rank-k approximation of the matrix $A = U\Sigma V^T$, where U and V are real-valued orthogonal matrices and Σ is diagonal. Although the resulting decompositions achieve low reconstruction errors, in many applications they may be hard to interpret. Nonnegative matrix factorization (NMF) (see [1] and references therein) is an attempt to overcome this problem by restricting the factor matrices to be nonnegative and avoiding the restriction to orthogonal matrices.

General CX and CUR decompositions have been studied previously in the fields of numerical linear algebra and theoretical computer science. A very recent and comprehensive study can be found in [7]. From the two algorithms used here 844 stems from the field of numerical linear algebra, while DMM stems from theoretical computer science.

6. CONCLUSIONS

We have introduced the nonnegative CX and CUR problems and proposed algorithms for solving them. Via an extensive set of experiments on synthetic and real datasets we have demonstrated that the decompositions we propose give naturally interpretable factors with very low reconstruction errors. More specifically, our algorithms for NNCX and NNCUR achieve reconstruction errors that are consistently smaller than the errors achieved by the nonnegative versions of the 844, DMM and Kmeans algorithms. More importantly, in many cases, our obtained solutions give errors that are smaller or equal to the reconstruction errors obtained by 844, DMM and Kmeans even without the nonnegativity constraints. This implies that the decompositions we propose are natural and that our algorithms make good choices in the selection of the columns and rows in the factor matrices. This last statement is further supported by the fact that the NNCX and NNCUR decompositions achieve reconstruction errors very close to those of NMF and SVD, which are lower bounds on the errors one should expect.

Developing new algorithms for NNCX and NNCUR decompositions is of course the main direction for the future work. Also, the *model-selection* question, i.e., of how many columns one should select to C and how many rows to R, is an interesting direction for future research.

7. REFERENCES

[1] M. W. Berry et al. Algorithms and applications for approximate nonnegative matrix factorization. *Computational Statistics & Data Analysis*, 52(1):155–173, 2007.

[2] M. W. Berry, S. A. Pulatova, and G. W. Stewart. Algorithm 844: Computing sparse reduced-rank approximations to sparse matrices. *ACM Trans. Math. Softw.*, 31(2):252–269, 2005.

[3] A. Çivril and M. Magdon-Ismail. Finding maximum volume sub-matrices of a matrix. Technical Report 07-08, Department of Computer Science, Rensselaer Polytechnic Institute, 2007.

[4] S. C. Deerwester et al. Indexing by latent semantic analysis. *J. of the American Society for Information Science*, 41(6):391–407, 1990.

[5] P. Drineas, M. W. Mahoney, and S. Muthukrishnan. Subspace sampling and relative-error matrix approximation: Column-based methods. In *APPROX-RANDOM*, pages 316–326, 2006.

[6] P. Drineas, M. W. Mahoney, and S. Muthukrishnan. Subspace sampling and relative-error matrix approximation: Column-row-based methods. In *ESA*, pages 304–314, 2006.

[7] P. Drineas, M. W. Mahoney, and S. Muthukrishnan. Relative-error CUR matrix decompositions, Aug. 2007. arXiv:0708.3696v1 [cs.DS].

[8] S. M. Embleton and E. S. Wheeler. Finnish dialect atlas for quantitative studies. *J. of Quantitative Linguistics*, 4(1–3):99–102, 1997.

[9] S. M. Embleton and E. S. Wheeler. Computerized dialect atlas of Finnish: Dealing with ambiguity. *J. of Quantitative Linguistics*, 7(3):227–231, 2000.

[10] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151, 2001.

[11] G. H. Golub and C. F. Van Loan. *Matrix Computations*. JHU Press, 1996.

[12] M. K. Kozlov, S. P. Tarasov, and L. G. Hačijan. Polynomial solvability of convex quadratic programming. *Soviet Math. Dokl.*, 20(5):1108–1111, 1979.

[13] C. Papadimitriou and K. Steiglitz. *Combinatorial Optimization Algorithms and Complexity.* Dover Publications, 1998.