A divide-and-conquer algorithm for binary matrix completion [☆]

Melanie Beckerleg, Andrew Thompson [*]

Mathematical Institute, University of Oxford, Andrew Wiles Building, Woodstock Road, Oxford, OX2 6GG, UK.

A R T I C L E   I N F O

A B S T R A C T

We propose a practical algorithm for low rank matrix completion for matrices with binary entries which obtains explicit binary factors and show it performs well at the recommender task on real world datasets. The algorithm, which we call TBMC (*Tiling for Binary Matrix Completion*), gives interpretable output in the form of binary factors which represent a decomposition of the matrix into tiles. Our approach extends a popular algorithm from the data mining community, PROXIMUS, to missing data, applying the same recursive partitioning approach. The algorithm relies upon rank-one approximations of incomplete binary matrices, and we propose a linear programming (LP) approach for solving this subproblem. We also prove a 2-approximation result for the LP approach which holds for any subsampling pattern, and show that TBMC exactly solves the rank-k prediction task for a underlying block-diagonal tiling structure with geometrically decreasing tile sizes, providing the ratio between successive tiles is less than $1/\sqrt{2}$. Numerical experiments show that TBMC outperforms existing methods on real datasets.

© 2020 Elsevier Inc. All rights reserved.

1. Introduction

1.1. Matrix completion

Matrix completion is an area of great mathematical interest and has numerous applications including recommender systems for e-commerce, bioactivity prediction and models of online content, such as the famous Netflix problem.

The recommender problem in the e-commerce setting is the following: given a database where rows are users and column entries indicate user preferences for certain products, fill in the entries of the database so as to be able to recommend new products based on the preferences of other users. Typically these matrices are highly incomplete, since most users will only have experienced a small fraction of all available products [2]. Similarly, in bioactivity prediction, compound-protein interaction (CPI) databases record bioactivity between potential drug compounds (rows, or users) and target proteins (columns, or products). Obtaining experimental evidence of all possible interactions is prohibitively expensive however, and therefore there are few known entries relative to the overall size of the database, see for example [20].

Low rank approaches to matrix completion have been the focus of a great deal of theoretical and algorithmic exploration ever since the seminal work in [9,7]. In many cases, the problem is formulated as follows. Given a database $\mathbf{A} \in \mathbb{R}^{m \times n}$, with observed entries in Ω, find a matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$ with minimal rank that matches the observed entries up to a given tolerance, i.e. which solves

$$\min ||\mathcal{P}_\Omega(\mathbf{A} - \mathbf{X})||_2^2 \text{ s.t. } \text{rank}(\mathbf{X}) \leq r \tag{1}$$

where r is some small integer and $\mathcal{P}_\Omega$ is the projection to the space of known entries, such that the error is evaluated only for the $(i, j) \in \Omega$.

A variety of algorithms have been proposed to solve this non-convex problem, see [28] and references therein. Another popular approach is to solve a convex relaxation involving the nuclear norm [7]. Such algorithms have been applied successfully in a wide range of applications, including recommender systems [15].

1.2. Binary matrix completion

The recommender problem can be seen as a binary decision problem. When applied to a binary matrix, $\mathbf{A} \in \mathbb{B}^{m \times n}$, the output of the matrix completion algorithms described above cannot be guaranteed to be binary, and a typical approach for matrix completion with binary data is to threshold the output from an algorithm for completion with real data [29,3]. As highlighted in [31], where databases follow model assumptions and recovery from solving is exact, factorisation reduces to assigning identical rows to the same cluster. However, this approach is not robust to the violation of assumptions and

breaks down when recovery is not exact. This motivates the search for algorithms which explicitly seek binary solutions.

The problem (1) can also be formulated as one of approximating $\mathbf{X}$ as the product of factors, namely

$$\min ||\mathcal{P}_\Omega(\mathbf{A} - \mathbf{U}\mathbf{V}^T)||_2^2 \text{ s.t. } \mathbf{U} \in \mathbb{R}^{m \times k}, \mathbf{V} \in \mathbb{R}^{n \times k} \ . \tag{2}$$

We are interested in this paper in approximating our database with binary factors, due to the greater interpretability of the output. To appreciate this, note that a binary factorisation

$$\mathbf{U}\mathbf{V}^{\mathbf{T}} = \sum_{i=1}^{k} \mathbf{U}_{:,i} \mathbf{V}_{:,i}^{T}$$

decomposes a binary matrix into biclusters of rows and columns, often referred to in the itemset mining community as *tiles*, see for example [10]. This decomposition provides an explicit characterization of the row/column clusters that best explain the database. Low rank decompositions designed for real matrices, on the other hand, tend to be SVD-like and orthogonal, with negative entries, and it is less clear how to interpret these. Low rank matrix completion with nonnegativity constraints, for example in [33], enforces non-negativity of factors, but does not address the issue of rounding errors induced by rounding non-integer values. Matrix completion algorithms were proposed in [32,30] which obtain decompositions in which one factor is composed of rows of the matrix, and is by consequence binary, although the other factor is generally non-integer.

However, in the case where both factors are required to binary, research to date has largely focused on the case of *binary matrix factorisation* (BMF) in which the matrix is fully observed. BMF has found applications in itemset mining of transactional and textual data [10], and also in the analysis of gene expression data [35]. In these applications, entries are in general fully observed, though possibly with noise.

Various algorithms for BMF have been proposed. The problem can be formulated as an integer program, but the use of an integer programming solver is only tractable for very small problem sizes. An approach combining convex quadratic programming relaxations and binary thresholding was proposed in [35]. For the special case of rank-one approximation, linear programming relaxations were proposed in [27,21,35], the first two of which also both proved that the linear programming solution yields a 2-approximation to the optimal objective. An approach to binary factorisation based on k-means clustering was considered in [13]. A local search heuristic capable of improving solutions obtained by other methods was proposed in [25]. Of particular relevance to this paper is the PROXIMUS algorithm, proposed in [17], which identifies patterns within the data using recursive partitioning based on rank-one approximations.

Closely related to BMF is the *Boolean* matrix factorisation problem, in which $\mathbf{U}\mathbf{V}^T$ is replaced by the OR operator, $\mathbf{U} \wedge \mathbf{V}$, which allows the tiles to overlap. A number of algo-

rithms also exist for Boolean factorisation, including the iterative ASSO algorithm [24] and integer programming [16].

1.3. Our contribution and related work

The novelty of this paper is the use of a partitioning approach to solve the problem of BMF with missing data, which can be formulated as follows. For $\mathbf{A} \in \mathbb{B}^{m \times n}$, solve

$$\min_{\substack{\mathbf{U} \in \mathbb{B}^{m \times r} \\ \mathbf{V} \in \mathbb{B}^{n \times r}}} ||\mathcal{P}_{\Omega}(\mathbf{A} - \mathbf{U}\mathbf{V}^T)||_2^2 . \tag{3}$$

The contributions of this paper are as follows.

- We propose TBMC (Tiling for Binary Matrix Completion), a low rank binary matrix completion algorithm (Section 2). The algorithm extends the recursive partitioning approach of [17] for BMF by means of rank-one approximations.
- In particular, we propose using an LP rank-one approximation for missing data. We support this choice with a guarantee that it provides a 2-approximation to the optimal objective value, showing that the reasoning of [27] holds in the missing data case (Section 3).
- We show that, under the assumption of a block diagonal model for our data, this algorithm correctly identifies tiles for a rank-k database with geometrically decreasing tiles, where the ratio between successive tiles is less than $1/\sqrt{2}$.
- We show that our algorithm outperforms alternatives based on related heuristics and techniques for non-negative matrix completion and binary matrix completion, when tested on synthetic and real life data (Section 5).

The most closely related work we are aware of is the *Spectral method* proposed in [31] for bi-clustering databases with missing data. The authors use low rank completion to cluster neighbour rows and then redefine the column clusters based on cluster membership, in a similar fashion to k-means. We show that our algorithm outperforms the Spectral method when solving Problem (3) for real world datasets.

The authors of [34] extend the ASSO algorithm for Boolean matrix factorisation to deal with missing data. However, it is worth pointing out that their setup, as well as solving a different problem, is primarily intended for only small amounts of missing data. Methods that involve linear programs can be straightforwardly extended to the missing data case, by evaluating the objective and enforcing constraints only for $(i, j) \in \Omega$, however we are not aware of any attempts to analyse their predictive power for the recommender problem.
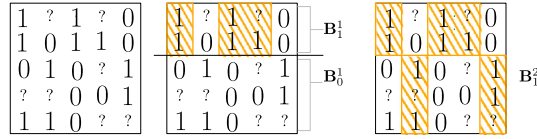
Fig. 1. A row-wise partition is formed from a rank-one approximation (a tile) by splitting according to which rows are included in the tile.

1.4. Relation to graph-theoretic problems

Viewing the database as the adjacency matrix of a graph, we can view (3) as a clustering problem; in particular the problem is that of approximating the edge set of a partially observed graph as a union of bicliques. The factors $\mathbf{U}$ and $\mathbf{V}$ can be interpreted as indicating the rows and columns of these bicliques. The authors of [29] solve the related problem for the diagonal block model, expressing the database as a sum of a low rank matrix plus a sparse component to account for noise. Our approach differs as we allow column overlap of clusters. Bi-clustering approaches, in particular for clustering gene expression data, have been used to solve formulations similar to the BMF problem, with different assumptions about the underlying data, equivalent to considering different constraints on $\mathbf{U}$ and $\mathbf{V}$. However these have focused primarily on cluster recovery; our focus is on exploring the predictive power of different algorithms for solving Equation (3).

2. The TBMC algorithm

We present an algorithm for low rank binary matrix completion inspired by the partitioning approach of [17] that generates a binary factorisation based on recursive rank-one partitions.

2.1. Partitioning

Our algorithm is inspired by the recursive partitioning approach of the PROXIMUS algorithm for binary matrix factorisation [17]. For a given submatrix, $\mathbf{B}$, the algorithm first calculates a binary rank-one approximation $\{\mathbf{u}, \mathbf{v}\}$. The matrix $\mathbf{B}$ is then partitioned based upon the rows included in the tile $\{\mathbf{u}, \mathbf{v}\}$: if the ith entry of $\mathbf{u}$ is positive, then the ith row is included in $\mathbf{B}_1$, else it is included in $\mathbf{B}_0$. Both submatrices are added to the set of submatrices to be partitioned.

The vector $\mathbf{v}$ is used as the pattern vector for $\mathbf{B}_1$. If the scaled Hamming distance, defined as $H(\mathbf{v}, A_{i,:}) = \sum_{j:(i,j)\in\Omega}(v_{ij} \neq A_{ij})/\sum_{j:(i,j)\in\Omega} 1$, from $\mathbf{v}$ to any of the rows in $\mathbf{B}_1$ is less than some tolerance t, for $0 < t < 1$, then the tile $\mathbf{u}\mathbf{v}^T$ is included in the decomposition and $\mathbf{B}_1$ is not partitioned any further. In addition, if the rank-one approximation returns a row vector of 1s, the tile is added to the decomposition. This process is repeated on successive submatrices until convergence. The algorithm terminates when no partitions remain. An illustration of the splitting process can be seen in Fig. 1.

2.2. Rank-one approximation with missing data

The partitioning method outlined above relies on rank-one approximations of the database. We outline two methods for this sub-problem.

2.2.1. Approximate rank-one using a linear program

Inspired by the work of [27], we propose a linear programming relaxation of the rank-one approximation problem with missing data.

The rank-one problem can be formulated as an integer linear program for the case where no data is missing [25,16]. The authors of [27] show that a linear program relaxation of a related problem has no more than twice the error of the integer solution. In Section 3 we show that a similar result holds for the missing data case.

We can formulate the best binary rank-one approximation problem as a linear program as follows:

$$
\begin{aligned}
\min_{\substack{\mathbf{u} \in \{0,1\}^m \\ \mathbf{v} \in \{0,1\}^n}} ||\mathcal{P}_\Omega(\mathbf{A} - \mathbf{u}\mathbf{v}^T)||_2^2 &= \min_{\substack{\mathbf{u} \in \{0,1\}^m \\ \mathbf{v} \in \{0,1\}^n}} \sum_{(i,j)\in\Omega} (A_{ij} - u_i v_j)^2 \\
&= \min_{\substack{\mathbf{u} \in \{0,1\}^m \\ \mathbf{v} \in \{0,1\}^n}} \sum_{(i,j)\in\Omega} A_{ij}^2 - 2(A_{ij} - 1)u_i v_j \ .
\end{aligned}
\tag{4}
$$

Since A is fixed, (4) is equivalent to solving

$$
\max_{\substack{\mathbf{u} \in \{0,1\}^m \\ \mathbf{v} \in \{0,1\}^n}} \sum_{(i,j):A_{ij}=1} u_i v_j - \sum_{(i,j):A_{ij}=0} u_i v_j.
\tag{5}
$$

We translate Problem (5) to a linear program as follows: introducing dummy variables z_{ij} which serve as indicator variables for $u_i v_j$ we can rewrite (5) as

$$
\begin{aligned}
\max_{\substack{\mathbf{u} \in \{0,1\}^m \\ \mathbf{v} \in \{0,1\}^n \\ \mathbf{z} \in \{0,1\}^{n\times m}}} & \sum_{(i,j)\in\Omega:A_{ij}=1} z_{ij} - \sum_{(i,j)\in\Omega:A_{ij}=0} z_{ij} \\
\text{s.t.} \quad & \left.\begin{aligned} u_i + v_j - z_{ij} &\leq 1 \\ 2z_{ij} &\leq u_i + v_j. \end{aligned}\right\} (i,j) \in \Omega
\end{aligned}
\tag{6}
$$

If we relax the integer constraints, then for optimal z_{ij}, the second constraint will be satisfied as an equality for $A_{ij} = 1$, so we can drop the corresponding dummy variables and replace their value in the objective with $\frac{1}{2}(u_i + v_j)$. Thus we consider the following formulation for approximating the solution to Equation (6)

$$
\begin{aligned}
\max_{\substack{\mathbf{u} \in \mathbb{R}^m \\ \mathbf{v} \in \mathbb{R}^n}} & \sum_{(i,j)\in\Omega:A_{ij}=1} \frac{1}{2}(u_i + v_j) - \sum_{(i,j)\in\Omega:A_{ij}=0} z_{ij} \\
\text{s.t.} \quad & \left.\begin{aligned} u_i + v_j - z_{ij} &\leq 1 \\ 0 < z_{ij} &< 1 \end{aligned}\right\} (i,j) \in \Omega : A_{ij} = 0 \\
& 0 < u_i, v_j < 1 \ (i,j) \in \Omega
\end{aligned}
\tag{7}
$$

The corresponding constraint matrix is totally unimodular (as it is a submatrix of the constraint matrix in [27]), and so solving will give integral values for $\{u, v, z\}$. We can obtain the corresponding rank-one approximation as $\mathbf{u}\mathbf{v}^T$. Note that z_{ij} is only defined for $(i, j) : A_{ij} = 0$. The remaining z values can be calculated as $1/2(u_i + v_j)$ but are no longer guaranteed to be integers.

2.2.2. Alternating minimisation

Alternating minimisation has been widely used for matrix completion, see for example [12]. Given a fixed $\mathbf{v}$ we want to find $\mathbf{u}$ to minimise the approximation error

$$||\mathcal{P}_\Omega(\mathbf{A} - \mathbf{u}\mathbf{v}^T)||_2^2 = \sum_{i,j\in\Omega} A_{ij}^2 - u_i(2A_{ij}v_j - v_j^2) \tag{8}$$

where we have used the fact that $\mathbf{u}$ and $\mathbf{v}$ are binary. Writing

$$W_{ij} = \begin{cases} 2A_{ij} - 1 & \text{if } i, j \in \Omega \\ 0 & \text{otherwise,} \end{cases} \tag{9}$$

we can write the right-hand side of (8) as

$$\sum_i \sum_{j:(i,j)\in\Omega} A_{ij}^2 - u_i W_{ij} v_j. \tag{10}$$

Suppose we are at iteration k and we are fixing $\mathbf{v}^k$. Then it follows from (10) that the updated binary $\mathbf{u}^{k+1}$ is given by

$$u_i^{k+1} = \begin{cases} 1 & \text{if } (\mathbf{W}\mathbf{v}^k)_i > 0 \\ 0 & \text{else.} \end{cases} \tag{11}$$

Then $\mathbf{v}^{k+1}$ can be calculated in a similar way. Typically the process converges in only a few iterations.

The authors of [17] outline a number of heuristics for initialising the iterative scheme. In particular, if not initialised correctly, this method can lead to the empty tile, which is always sub-optimal. We propose using alternating minimisation as an optional post-processing step for the TBMC algorithm stated in Section 2.3.

2.3. Statement of algorithm

Using the ideas of Section 2.1 and Section 2.2 we now propose an algorithm for binary matrix completion for recommender systems.

Starting with the full database, we calculate $\{\mathbf{u}^*, \mathbf{v}^*\}$ the rank-one approximation obtained by solving Problem (7) for the current partition, then partition into the rows that are included in the tile and those that are not, and repeat for both sides of the

partition. As in [17], when the partitioning process generates rows that are all within a given radius of $\mathbf{v}^*$ or when $\mathbf{u}^*$ is the vector of all 1s or all 0s, then $\{\mathbf{u}^*, \mathbf{v}^*\}$ is added to the factorisation. The algorithm is stated in detail in Algorithm 1. We could also consider using the alternating minimisation outlined in Section 2.2.2 to improve the approximation found by solving (7). We refer to this approach as TBMC-AM. In its current form, the algorithm only partitions by rows. We also considered a variant by which each tile identified by the algorithm was then partitioned in the same manner column-wise. It is possible to construct examples for which this would improve the accuracy but as we found no change in the output for any of our experiments, we speculate that these examples are rarely found in practise.

Algorithm 1 Tiling for Binary Matrix Completion (TBMC).

1: Initialise $S = \{\mathbf{A}\}$, $\mathbf{U} = \emptyset$, $\mathbf{V} = \emptyset$, $i = 1$
2: **while** $S \neq \emptyset$, $k < k_{max}$ **do**
3:     select $\mathbf{B} \in S$, set $S \leftarrow S \backslash \mathbf{B}$
4:     **procedure** FIND RANK-ONE APPROXIMATION FOR $\mathbf{B}$:
5:         Find $\{\mathbf{u}, \mathbf{v}\}$ as the solution to 7
6:     **procedure** PARTITION $\mathbf{B}$
7:         Define $J = \left\{ j : \mathbf{u}^k(j) = 1 \right\}$
8:         Set $\mathbf{B}_1 = \mathbf{B}(J, :)$, $\mathbf{B}_0 = \mathbf{B}(\overline{J}, :)$
9:     **procedure** UPDATE
10:         **if** $\mathbf{B}_0 \neq \mathbf{0}$ and $\mathbf{u} \neq \mathbf{0}$ **then** $S \leftarrow B_0$
11:         **if** $\max_j ||\mathbf{B}_1(j, :) - v^k|| < t$, or $\mathbf{u} = \mathbf{1}$
          then
                $U \leftarrow \mathbf{u}^k, V \leftarrow \mathbf{v}^k$, $k \rightarrow k + 1$
12:         **else** $S \leftarrow S \cup \mathbf{B}_1$

2.4. Algorithm complexity

The most computationally demanding step in Algorithm 1 is the solution of (7). To solve a non-degenerate LP with a simplex method, the worst case complexity is exponential in the number of variables (equal to $N = n + m + |\{(i, j) : (i, j) \in \Omega, A_{ij} = 0\}|$) and the number of constraints (equal to $M = 2N + |\{(i, j) : (i, j) \in \Omega, A_{ij} = 0\}|$), since we have one constraint for each z_{ij}, plus constraints that each variable be between 0 and 1. This upper bound arises because the method will terminate when it has checked every vertex of the feasible region defined by the problem. There are $M + N$ possible vertices, so there are 2^{M+N} possible intersections. However, since the constraint matrix of (7) is totally unimodular (see Section 2.2), a polynomial bound in M and N can be obtained. By Corollary 4.1 of [14], solving (7) using Bland's rule [5] (or another method for avoiding cycles) will terminate in at most $2N(M \log N + 1)$ operations.

The other steps in the algorithm have lower complexity. The alternating minimisation scheme requires one left and one right matrix vector multiplication per iteration, hence, since we impose a limit on the number of iterations, it takes $\mathcal{O}(n+m)$ steps. Calculation of the error (step 11) requires at most nm operations.

In order to determine the complexity of the algorithm as a whole, we multiply the complexity of the rank-one case by m, the worst case for the number of partitions that need to be checked. So Algorithm 1 requires at most $\mathcal{O}(mNMlogN)$ operations, i.e. polynomial in m and n.

We note that solving the problem directly involves checking k options for the membership of each row and column, i.e. k^{m+n} possibilities. Hence Algorithm 1 offers a significant improvement over solving the problem directly.

3. Approximation bounds for the rank-1 subproblem

In Section 2.2, we proposed a linear program method for solving the rank-one step of (1). By closely following the approach in [27] for the case in which all the entries in $\mathbf{A}$ are known, we show that the approximation error of the tile found by solving Problem (7) is no more than twice the optimal.

Theorem 1. *Denoting by $(\mathbf{u}^*, \mathbf{v}^*, \mathbf{z}^*)$ the optimal solution for Problem (7) obtained using a simplex method, then the approximation error $E = ||\mathcal{P}_\Omega(\mathbf{A} - \mathbf{u}^*\mathbf{v}^{*T})||_2^2$ satisfies*

$$E \leq 2 \min_{\substack{\mathbf{u} \in \{0,1\}^m \\ \mathbf{v} \in \{0,1\}^n}} ||\mathcal{P}_\Omega(\mathbf{A} - \mathbf{u}\mathbf{v}^T)||_2^2 \tag{12}$$

Proof. The minimal error for a single tile approximation is given by

$$
\begin{aligned}
\min_{\substack{\mathbf{u} \in \{0,1\}^m \\ \mathbf{v} \in \{0,1\}^n}} & \ ||\mathcal{P}_\Omega(\mathbf{A} - \mathbf{u}\mathbf{v}^T)||_2^2 \\
&= \min_{\substack{\mathbf{u} \in \{0,1\}^m \\ \mathbf{v} \in \{0,1\}^n}} \sum_{(i,j) \in \Omega} (A_{ij} - u_i v_j)^2 \\
&= \sum_{(i,j) \in \Omega} A_{ij}^2 - \max_{\substack{\mathbf{u} \in \{0,1\}^m \\ \mathbf{v} \in \{0,1\}^n}} \left(\sum_{(i,j):A_{ij}=1} u_i v_j - \sum_{(i,j):A_{ij}=0} u_i v_j \right) \\
&\geq \sum_{(i,j) \in \Omega} A_{ij}^2 - \left(\frac{1}{2} \sum_{(i,j):A_{ij}=1} (u_i^* + v_j^*) - \sum_{(i,j):A_{ij}=0} z_{ij}^* \right)
\end{aligned}
\tag{13}
$$

where the inequality is a result of the fact that the optimal solution for Problem (6) is bounded above by the objective for Problem (7) since relaxing constraints does not decrease the value at optimality.

For (i,j) such that $A_{ij} = 0$, we know that u_i, v_j and z_{ij} are integers. Since the objective of (7) is to minimise z_{ij}^*, which is bounded below by $u_i^* + v_j^* - 1$, we have that $z_{ij}^* = 1$ if and only if $u_i^* + v_j^* = 2$. Thus we can split the summation terms on the right hand side of the inequality to obtain

$$LHS \geq \sum_{(i,j) \in \Omega} A_{ij}^2 - \sum_{\substack{(i,j): \ A_{ij}=1 \\ u_i+v_j=2}} 1 - \sum_{\substack{(i,j): \ A_{ij}=1 \\ u_i+v_j=1}} \frac{1}{2} + \sum_{\substack{(i,j): \ A_{ij}=0 \\ u_i+v_j=2}} 1 \tag{14}$$

Since A is binary, we have that

$$\sum_{(i,j)\in\Omega} A_{ij}^2 \geq \sum_{\substack{(i,j)\,:\,A_{ij}=1\\u_i+v_j=1}} 1 + \sum_{\substack{(i,j)\,:\,A_{ij}=1\\u_i+v_j=2}} 1 \qquad (15)$$

which we can use to replace the sum in (14) corresponding to values where exactly one of u_i and v_j is 1 to derive

$$\begin{aligned}
\min_{\substack{\mathbf{u}\,\in\,\{0,1\}^m\\\mathbf{v}\,\in\,\{0,1\}^n}} ||\mathcal{P}_\Omega(\mathbf{A}-\mathbf{u}\mathbf{v}^T)||_2^2 &\geq \frac{1}{2}\left(\sum A_{ij}^2 - \sum_{\substack{(i,j)\,:\,A_{ij}=1\\u_i+v_j=2}} 1\right) + \sum_{\substack{(i,j)\,:\,A_{ij}=0\\u_i+v_j=2}} 1 \\
&\geq \frac{1}{2}\left(\sum A_{ij}^2 - \sum_{\substack{(i,j)\,:\,A_{ij}=1\\u_i+v_j=2}} 1 + \sum_{\substack{(i,j)\,:\,A_{ij}=0\\u_i+v_j=2}} 1\right)
\end{aligned} \qquad (16)$$

Now since $2A_{ij}-1$ is equal to 1 if A_{ij} is positive and -1 if A_{ij} is zero we can rewrite (16) as

$$\begin{aligned}
\min_{\substack{\mathbf{u}\,\in\,\{0,1\}^m\\\mathbf{v}\,\in\,\{0,1\}^n}} ||\mathcal{P}_\Omega(\mathbf{A}-\mathbf{u}\mathbf{v}^T)||_2^2 &\geq = \frac{1}{2}\sum_{(i,j)\in\Omega}\left(A_{ij}^2 - (2A_{ij}-1)\,u_iv_j\right) \\
&= \frac{1}{2}\sum_{(i,j)\in\Omega}(A_{ij}-u_iv_j)^2 = \frac{1}{2}E \ .
\end{aligned} \qquad (17)$$

This gives us our bound.   □

Note that in the case where the true best solution has zero error, the LP relaxation will also have zero error, and therefore for a database consisting of a single planted tile, solving (7) will recover this tile provided that, for each row, the sampling operator sees a positive and a negative entry for each row or column.

3.1. Verifying Theorem 1

We first perform experiments to illustrate numerically that the 2-approximation result given in (12) is not violated.

We evaluate performance relative to the optimal solution, which we obtain by solving (6) directly as an integer program. We generate binary matrices with (a) a single planted tile and (b) 3 planted tiles, with τn positive entries per row in each tile. We simulate noisy data by randomly flipping a fraction ϵ of entries and remove a proportion $1-\rho$ of entries. We set $\tau = 0.7$ and $\epsilon = 0.03, \rho = 0.7$. Note that choice of parameters is to provide an illustrative example, as we observe similar behaviour for other parameter configurations. To make it tractable to solve the problem directly, we consider databases of size 100×100 for the single tile case and 10×10 for the 3 tile case, since the latter

Table 1

Mean value of $\mathcal{R}$ and P_0, the proportion of cases test cases for which $\mathcal{R} > 2$, for matrices with a single planted tile, $m = 100$.

	Mean $\mathcal{R}$	P_0
LP (AM)	1.0 **(1.0)**	0 **(0)**
NMF (AM)	0.0 **(0.0)**	0.0 **(0.0)**
Partition (AM)	2.42 (1.26)	0.87 **(0.0)**
Avg (AM)	2.56 (1.32)	0.62 (0.0)

Table 2

Mean value of $\mathcal{R}$ and proportion of cases test cases for which $\mathcal{R} > 2$, for matrices with 3 planted tiles with density $\tau = 0.7$ with $m = 10$.

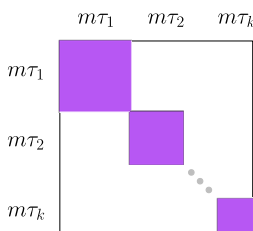|	Mean $\mathcal{R}$	P_0
LP (AM)	1.6 **(1.04)**	0 **(0)**
NMF (AM)	2.46 (1.21)	0.43 (0.01)
Partition (AM)	2.94 (1.37)	0.58 (0.12)
Avg (AM)	2.96 (1.21)	0.74 (0.06)

Fig. 2. We consider recovery guarantees for case of a block diagonal model with decreasing tile size.

is more computationally intensive to solve. In both cases, we calculate the ratio, $\mathcal{R}$, of squared l_2 approximation error to the optimal squared l_2 error. The mean value of $\mathcal{R}$ and the proportion of cases for which $\mathcal{R}$ is greater than 1 is recorded in Section 3.1 (Tables 1, 2).

We solve the LP relaxation using the simplex method implemented by MATLAB's linprog solver [8]. In addition, we compare against other methods, *average, partition*, and *nmf* for generating a rank-1 approximation, further details of which can be found in Section 5. We observe that while the other methods all violate the 2-approximation bound for the 3-tile model, the LP does not.

4. Recovery guarantees for TBMC

The results of Section 3 hold for any binary matrix and give guarantees for solving the rank-1 problem. We now explore the conditions for which Algorithm 1 will recover a database generated according to a planted tile model. We consider the following model for the underlying data: let $\mathbf{A} \in \mathbb{B}^{m \times m}$ have a symmetric block diagonal structure, with the size of the lth tile equal to $(m\tau_l)^2$ where $\tau_{l+1} \leq \tau_l$ as illustrated in Figs. 2, 3.

The heart of the proof is in showing that, in the case of no missing data, solving Problem (7) recovers the largest tile. We then show that for the case of no missing data, if the algorithm recovers the first i tiles then it will recover the $(i+1)$th tile and conclude.

Fig. 3. On the ith iteration of Algorithm 1, empty columns can be ignored, as setting $\beta_j = 1$ for $j < i$ will never increase the objective.

Finally, we show numerically that the performance for the case of missing data is in close agreement with the theoretical result for no missing data. We define T_l to be the set of indices in the lth tile and

$$S_{l,l'} = \{(i,j) : i \in T_l, j \in T_{l'}\}.$$

We rely on the following observations:

Lemma 2. *We can rewrite the objective of (7) as*

$$f(\mathbf{u}, \mathbf{v}) = \sum_l \left(\sum_{\substack{(i,j)\,\in\,S_{l,l} \\ u_i + v_j = 1}} \frac{1}{2} + \sum_{\substack{(i,j)\,\in\,S_{l,l} \\ u_i + v_j = 2}} 1 \right) \tag{18}$$

$$- \sum_{l \neq l'} \sum_{\substack{(i,j)\,\in\,S_{l,l'} \\ u_i + v_j = 2}} 1.$$

Proof. The constraints of (7) enforce the condition that $z_{ij} = u_i v_j$. Hence, if we split the second term of the objective of Problem (7) by the values of $u_i + v_j$ the only terms that will have a contribution are the ones for which $u_i + v_j = 2$. The rest follows by splitting the summation terms according to the values of $u_i + v_j$. □

Lemma 3. *Let α_l, β_l be the proportion of rows and columns of tile l included in a feasible solution to (7). Then we can write (18) as*

$$f(\boldsymbol{\alpha}, \boldsymbol{\beta}) = m^2 \sum_l \tau_l^2 \left\{ \frac{1}{2} \left[\alpha_l(1 - \beta_l) + (1 - \alpha_l)\beta_l \right] + \alpha_l \beta_l \right\}$$

$$- m^2 \sum_{l=1}^k \sum_{\substack{l'=1 \\ l' \neq l}}^k \alpha_{l'} \beta_l \tau_{l'} \tau_l. \tag{19}$$

In addition, there is an optimal solution for which $\alpha_l, \beta_l \in \{0, 1\}$ for each tile.

Proof. This follows from Equation (18). The objective can be split into sums by row; rows in the same tile will have identical contribution to the objective, leading to the formulation in (19). Since this expression is linear in the α_l for each l, we can conclude that there must be an optimal solution for either $\alpha_l = 0$ or $\alpha_l = 1$; to determine which requires determining the sign of the coefficient of each α_l: if it is positive then $\alpha_l = 0$ is optimal, and if it is negative then $\alpha_l = 1$ is optimal. The same reasoning applies to each β_l. □

We analyse tile recovery for the case where all entries are known, obtaining a phase transition for recovery in terms of relative tile sizes, and then show that, in practice, we get agreement with this phase transition for sub-sampled matrices, provided the tiles are large enough.

Theorem 4. *For a binary matrix $\mathbf{A}$ with a symmetric block diagonal structure with k tiles, having size $m\tau_l^2$ where $\tau_{l+1} = a\tau_l$, for $a \leq 1/\sqrt{2}$, Algorithm 1 recovers the tiles exactly in the case where all entries are known (in this case Algorithm 1 reduces to the PROXIMUS algorithm [17]).*

Proof. Using Lemma 3, we may consider four cases depending on the integer values of α_1, β_1.

Case (i): For $\alpha_1 = \beta_1 = 1$, we calculate

$$\frac{1}{m^2}\frac{\partial f}{\partial \alpha_j} = \frac{1}{2}\tau_j^2 - \tau_j\tau_1 - \sum_{\substack{l \neq j \\ l > 1}} \tau_j\tau_l \quad \text{for } j > 1 \tag{20}$$

$$\leq \frac{1}{2}\tau_j^2 - \tau_j\tau_1$$

This is strictly negative, since $\tau_1 > \tau_j$, hence $\alpha_j = 0$ for all $j > 1$. Similarly, $\beta_j = 0$ for all $j > 1$. The corresponding objective value is $m^2\tau_1^2$.

Cases (ii) and (iii): For $\alpha_1 = 1$, $\beta_1 = 0$,

$$\frac{1}{m^2}\frac{\partial f}{\partial \beta_j} \leq \frac{1}{2}\tau_j^2 - \tau_j\tau_1 \quad \text{for } j > 1 . \tag{21}$$

Hence $\beta_j = 0$ for all $j > 1$. We can then substitute this to find

$$f(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{2}m^2\tau_1^2 \left(\tau_1^2 + \sum_{l > 1} \alpha_l\tau_l^2 \right). \tag{22}$$

So for optimality in this case, $\alpha_l = 1, \beta_l = 0$ for all l. By symmetry, for $\alpha_1 = 0, \beta_1 = 1$, optimality in this case is obtained for $\alpha_l = 0, \beta_l = 1$ for all l. In both cases, the objective value is $f = \frac{1}{2}m^2\sum_l \tau_l^2$.

Case (iv): For $\alpha_1 = 0$, $\beta_1 = 0$, the objective is at most $m^2 \sum_{l>1} \tau_l^2/2$. To see this, consider that the lth term of the first summation is increasing in α_l and β_l, so obtains its maximum for $\alpha_l = \beta_l = 1$, and the second summation is always negative.

We can now compare the objectives for each case. Case (i) is optimal over the other cases, provided

$$\sum_{l>1} \tau_1^2 > \frac{1}{2} \sum_l \tau_l^2, \tag{23}$$

which is true if and only if

$$\tau_1^2 > \sum_{l>1} \tau_l^2 . \tag{24}$$

Since this is a strict inequality, the optimal solution will be unique.

Note that on subsequent iterations of step 2 of Algorithm 1, the input matrix is no longer square, but has dropped out the rows corresponding to the largest i tiles. The resulting matrix contains the remaining $k - i$ tiles and empty columns corresponding to the dropped tiles. Hence we are no longer tracking the α_j for $j \leq i$. Therefore, we rewrite the objective as

$$f(\boldsymbol{\alpha}, \boldsymbol{\beta}) = m^2 \sum_{l=i+1}^{k} \tau_l^2 \left\{ \frac{1}{2} \left[\alpha_l(1 - \beta_l) + (1 - \alpha_l)\beta_l \right] + (1 - \alpha_l)(1 - \beta_l) \right\}$$
$$- m^2 \sum_{\substack{l=i+1 \\ }}^{k} \sum_{\substack{l'=i+1 \\ l' \neq l}}^{k} \alpha_{l'} \beta_l \tau_{l'} \tau_l - m^2 \sum_{l=1}^{i} \sum_{\substack{l'=i+1 \\ l' \neq l}}^{k} \alpha_{l'} \beta_l \tau_{l'} \tau_l \tag{25}$$

where we have dropped out the terms corresponding to the dropped rows, and isolated terms corresponding to the empty columns ($l \leq i$). In this case, the coefficient of β_j for $j \leq i$ will be non-positive, and negative provided at least one of the α_j is positive for $j > i$. This means we can consider optimality for the reduced matrix formed by dropping out the columns corresponding to the first i tiles and therefore if

$$\tau_i^2 > \sum_{l>i} \tau_l^2 \tag{26}$$

we have that at each iteration, the algorithm will successfully identify the right tile, and therefore 1 will output the exact tiling after k iterations.

In particular, if we consider the geometric model for tile size, $\tau_{l+1} = a\tau_l$, then imposing (26) leads to the condition

$$1 > \sum_{l=1}^{k-i} a^{2l} = a^2 \sum_{l=1}^{k-i} a^{2(l-1)} = a^2 \frac{1 - a^{2(k-i)}}{1 - a^2} \tag{27}$$

for all $0 \leq i \leq k - 1$, from which we obtain

$$2a^2 - a^{2(k-i+1)} < 1 \tag{28}$$

In particular, for $a \leq 1/\sqrt{2}$ this requirement is satisfied for all i. $\square$

4.1. Observed behaviour with missing entries

Now suppose that entries are erased independently with probability $0 \leq \rho < 1$. We are interested in whether the phase transition identified above can be extended to this case. For a given realisation, we can no longer consider each row as identical, hence the theoretical result in this section does not precisely extend. However, our numerical observations suggest that we do observe a behaviour which is a close approximation, provided the database size is large enough.

We generate symmetric block-diagonal matrices with decreasing tile size according to the model in Section 4, setting $k = 4$, for $a = 0.1 : 1$ and $\rho = 0.1$ to 0.9. For 100 random trials, we calculate the proportion of matrices recovered exactly from Algorithm 1 and plot the results in the top row of Fig. 4. We also plot, in the bottom row of Fig. 4, the proportion for which Algorithm 1 is able to recover all but 3% of the entries; accounting for the difficulty of recovery of the smallest tiles. According to (28), we would expect to see recovery for $a \leq 0.72$. In both cases, we see that there is agreement with our bound on a, and that as we increase the size of our database, the value of maximum ρ for which the algorithm recovers all of the tiles also increases. The small gap between the dashed line indicating $a = 0.72$ and the line of failure to recover is an artefact of discretisation error for smaller databases; it is not possible to generate tiles that are exactly a factor of 0.72 smaller than the previous, since a row is either included or not.

5. Numerical results

We next demonstrate that the partitioning approach, and in particular our TBMC algorithm is a practical method for generating interpretable rules for inferring missing information in real data sets (recommender systems), that outperforms state of the art alternative methods for binary matrix completion.

5.1. Other algorithms

We compare Algorithm 1 to other methods for rank-k approximation: two alternative heuristics for partitioning similar to those used in [17]; NMF with missing values implemented using the NMF MATLAB Toolbox [26], based on the multiplicative scheme of [19] with zero weighting of missing entries; and a method that follows a k-means approach to cluster the observed data based on projections onto the k largest singular vectors of the observed data. In particular we consider
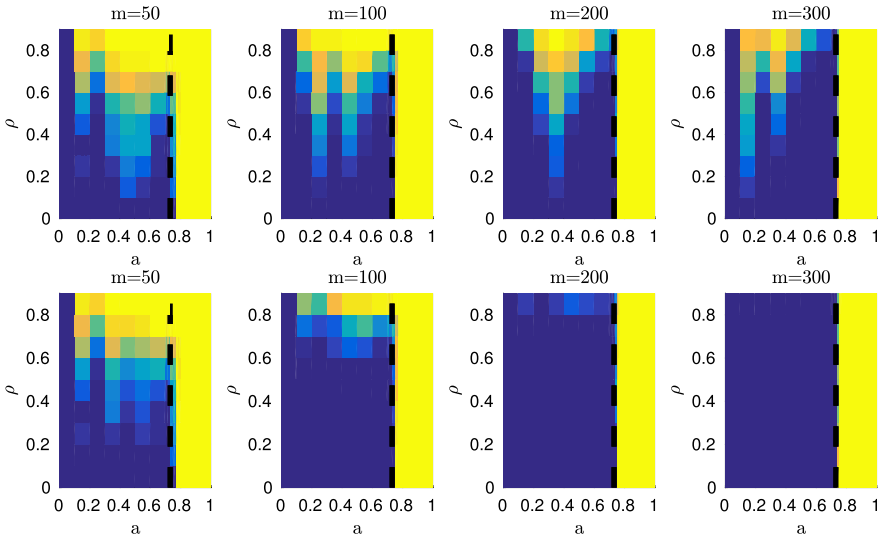
Fig. 4. Proportion of matrices, generated according to Section 4, which are (top) recovered by TBMC and (bottom) for which 97% of the entries are recovered (bottom). Dark blue regions are fully recovered. The black dashed line indicates the theoretical bound on a. (For interpretation of the colours in the figure(s), the reader is referred to the web version of this article.)

- *'average'*: setting $v_j = 0$ where the jth column has more than half of its entries as positive and $u_i = 1$ if any of the entries of the ith row have a positive entry in one of those columns;

- *'partition'*: selecting a column at random to be **u** and calculating **v** as the average of all the rows that have positive entries in that column, binarised with a threshold of $1/2$;

- *'NMF'*: a non-negative rank-one factorisation is obtained using a multiplicative update scheme with missing data and the factors are binarised using a threshold of 0.5. We use the approach outlined in [35] to normalize the factors such that the 0.5 threshold is appropriate.

- *'Spectral'*: We obtain a rank-one approximation by implementing the method outlined in [31], which follows a k-means approach. Briefly, the factors are initially generated using a projection onto the first k singular values; viewing the factorisation as a clustering, the footprints in $\mathbf{V}_i$ are updated using the clustering given by U_i, before reassigning rows to the cluster whose footprint most closely matches their own.

For TBMC, *average* and *partition* we also consider using an alternating minimisation step, outlined in Section 2.2.2 to improve the rank-one approximations. We refer to these approaches as TBMC-AM, *average*-AM and *partition*-AM.

5.2. Real world datasets

We consider performance on a series of real world recommender systems. It is worth pointing out that whilst many of the relevant datasets have non-binary entries, often categorical ratings for example, the aim of the recommender problem is to decide whether or a particular entry will be a positive interaction or not (i.e. whether a user will like a particular film, or a drug will bind to a particular target). Hence it is necessary to make decisions about how these datasets are binarised.

In all of the datasets we consider, we find optimum performance for partition based methods, and in particular for four of the five data sets that we investigate, we find that TBMC outperforms other state of the art methods upon this task.

We consider the performance of rank-k binary matrix factorisation on the following datasets.

- ChEMBL: Data of protein-ligand interactions, obtained from ChEMBL [4], version 21. Assay values for measurement types POTENCY, IC50, KI, MIC, EC50, KD, AC50 were binarised using a threshold of 2μM to obtain positive and negative activity categories, where negative represents no discernible effect on protein behaviour. The resulting matrix was then filtered by counting the positive and negative interactions, ranking the compounds based on the total number of interactions and taking subsets from the top 50000 ligands.
- ML: MovieLens 100k [22], which contains $100,000$ ratings from 943 users across 1682 films. We threshold to consider only 4 or 5 star ratings as positive.
- RC: customer reviews from 138 customers for 130 restaurants [23]. Since ratings are in $\{1, 2, 3\}$, we map to binary data by converting 3 star ratings to one, and converting 1 or 2 star ratings to zero.
- ALL-AML (GE): gene expression data [11], containing information from human tumour samples of different leukemia types. This dataset has been widely used for cancer classification, and for the evaluation of BMF techniques in the case of no missing data [6]. We normalize the data by column.
- Netflix: a subset of data obtained from Kaggle [18] from the Netflix prize data [1] of ratings by users on different movies. The sparsely reviewed movies (< 2000 ratings) and sparsely active users (< 52 reviews) are removed. The resulting dataset reflects the 70th percentile for density of reviews. In order to apply the algorithms below, we consider a random subset of 1000 users and 1000 movies. We threshold to consider only 4 or 5 star ratings as positive.

We compare the performance of TBMC and TBMC-AM against NMF and the Spectral method (see Section 3.1). The rank parameter was optimised for NMF, with 5 being found to be a good choice. The maximum rank was set to be 5 in the Spectral method, although the method was found to often terminate before five tiles were found. Although increasing the rank initially leads to an improve in approximation accuracy, none of these

methods are guaranteed to find a zero error approximation simply by increasing the rank. In addition, there is a risk of over-fitting for a rank parameter that is too high. We also compare with a variant of the recursive partitioning approach of TBMC in which the LP-based rank-one approximation step is replaced by the *average* and *partition* heuristics (see Section 3.1), again with and without alternating minimisation.

5.3. Performance of TBMC for rank-k decomposition

We consider a 70/30 split between known training entries and unseen test entries ($\rho = 0.7$) and set $t = 0.05$, although we find the impact of changing t has little impact below 0.1. Note that MovieLens, RC, ChEMBL and Netflix have a low density of known entries ρ_D. In all cases we record the percentage approximation error $\mathcal{P}$. We give approximation errors upon both the test set, and also upon the training set, averaged over 100 random trials. The first score tells us how well each method is able to predict missing entries. The second score tells us how well the method is able to generate a tiling model for the known data.

From Table 3, we see that TBMC outperforms all other algorithms as a predictive method on all five data sets. The alternating minimisation (AM) step leads to improved results in some cases, but not always. The improvement over other methods is significant in the case of the gene expression (GE) and restaurant customer (RC) data. On the other hand, in the case of the MovieLens data, the improvement gained by using TBMC over the NMF method is marginal and for both MovieLens and Netflix, the gain over simpler partitioning using simpler heuristics, *average* and *partition* heuristics is also marginal.

Approximation error on the training set allows us to compare how well the database tilings of the various methods are able to capture the known data. We see from Table 3 that TBMC outperforms all other methods on this task for the RC and GE datasets. For MovieLens, replacing the LP rank-one approximation in TBMC with the averaging heuristic is seen to give the best performance while for ChEMBL *partition* gives the best performance. These results support the use of a partitioning for the approximation task.

We speculate that the varying results for different datasets are due in large part to the degree to which each database can be modelled as a low-rank binary factorisation (tiling). It should be emphasised that the prediction algorithms considered here are restricted to those which generate interpretative binary factorisations. We are not claiming that TBMC competes favourably with all prediction algorithms (such as neural networks) if the interpretability requirement is removed. That said, our results show clearly that partitioning algorithms, and in particular TBMC, perform well compared to other algorithms for low-rank matrix completion with binary factors.

We conclude by making two further observations. Firstly, the fact that simple heuristics can perform well highlights the importance of balancing higher accuracy with constraints on computational time. Secondly, the algorithm shows evidence of over-fitting the RC data as the percentage error on the known values (training set) is 5.5%.

Table 3
Proportional error $\mathcal{P}$ for different datasets, on both test and training sets, with and without alternating minimisation (AM).

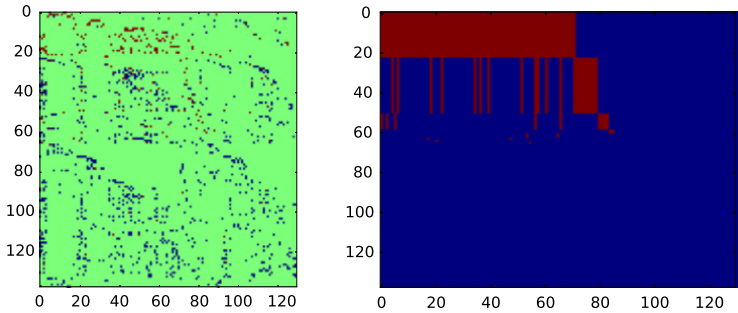|					$\mathcal{P}$ Test (AM) / Train (AM)				
	m	n	ρ	ρ_D	TBMC	Avg	Partition	NMF	Spectral
ChEMBL	7500	2183	0.7	0.01	19.0 **(15.4)**	30.7 (16.0)	29.7 (17.1)	27.7	29.9
					16.5 **(14.2)**	18.0 (18.0)	15.7 (14.7)	26.4	30.7
ML	1600	943	0.7	0.06	**20.8** (22.7)	21.3 (23.3)	33.8 (30.1)	21.1	21.1
					20.6 (19.5)	**17.7** (19.0)	33.3 (23.1)	19.5	21.3
RC	138	130	0.7	0.065	**19.5** (35.5)	22.2 (31.4)	23.3 (29.7)	22.2	22.0
					4.8 (5.5)	17.2 (18.2)	23.4 (17.7)	16.1	22.2
GE	5000	38	0.7	1.0	11.6 **(10.9)**	21.2 (21.2)	20.4 (15.2)	14.3	21.2
					11.3 (10.4)	21.5 (21.5)	18.7 (12.2)	12.6	21.5
Netflix	1000	1000	0.7	0.06	**18.3** (22.8)	18.9 (21.0)	19.9 (18.6)	20.4	19.9
					15.7 (13.57)	17.0 (13.9)	20.0 (14.2)	16.4	20.0

Fig. 5. The RC dataset (left) with rows and columns reordered according to the factorisation found by TBMC (right).

The RC database appears to be an especially good fit for approximation by binary factors (tiles). The clustering found by TBMC on the RC database is provided as an illustration in Fig. 5. The rows and columns have been reordered for visual effect.

6. Conclusion and future directions

Binary matrix completion for recommender systems has numerous applications. Influenced by approaches to binary matrix factorisation in the itemset mining literature, we have proposed the TBMC algorithm for binary matrix completion and shown that it outperforms alternatives on both synthetic and real data sets for certain regimes. These

results make the case for the consideration of heuristic methods where typical assumptions for low rank matrix completion are violated and exact recovery is not guaranteed.

We have presented a theoretical recovery guarantee for TBMC for geometrically decaying diagonal blocks. It would interesting to extend the result to incorporate random models for missing data and noise, and also to consider alternative block models.

Declaration of competing interest

The Authors declare that there is no competing interest.

References

[1] http://www.netflixprize.com.
[2] C.C. Aggarwal, Recommender Systems, Springer, 2016.
[3] B.P. Ames, S.A. Vavasis, Nuclear norm minimization for the planted clique and biclique problems, Math. Program. 129 (2011) 69–89.
[4] A.P. Bento, A. Gaulton, A. Hersey, L.J. Bellis, J. Chambers, M. Davies, F.A. Krüger, Y. Light, L. Mak, S. McGlinchey, et al., The chembl bioactivity database: an update, Nucleic Acids Res. 42 (2014) D1083–D1090.
[5] R.G. Bland, New finite pivoting rules for the simplex method, Math. Oper. Res. 2 (1977) 103–107.
[6] J.-P. Brunet, P. Tamayo, T.R. Golub, J.P. Mesirov, Metagenes and molecular pattern discovery using matrix factorization, Proc. Natl. Acad. Sci. 101 (2004) 4164–4169.
[7] E.J. Candès, B. Recht, Exact matrix completion via convex optimization, Found. Comput. Math. 9 (2009) 717.
[8] G.B. Dantzig, A. Orden, P. Wolfe, et al., The generalized simplex method for minimizing a linear form under linear inequality restraints, Pac. J. Math. 5 (1955) 183–195.
[9] M. Fazel, Matrix rank minimization with applications, PhD thesis, Stanford University, 2002.
[10] F. Geerts, B. Goethals, T. Mielikäinen, Tiling databases, in: International Conference on Discovery Science, 2004, pp. 278–289.
[11] T.R. Golub, D.K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J.P. Mesirov, H. Coller, M.L. Loh, J.R. Downing, M.A. Caligiuri, et al., Molecular classification of cancer: class discovery and class prediction by gene expression monitoring, Science 286 (1999) 531–537.
[12] P. Jain, P. Netrapalli, S. Sanghavi, Low-rank matrix completion using alternating minimization, in: Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing, ACM, 2013, pp. 665–674.
[13] P. Jiang, J. Peng, M. Heath, R. Yang, A clustering approach to constrained binary matrix factorization, in: Data Mining and Knowledge Discovery for Big Data, Springer, 2014, pp. 281–303.
[14] T. Kitahara, S. Mizuno, A bound for the number of different basic solutions generated by the simplex method, Math. Program. 137 (2013) 579–586.
[15] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer (2009) 30–37.
[16] R. Kovács, O. Günlük, R. Hauser, Low-rank Boolean matrix approximation by integer programming, in: NIPS Workshop on Optimization for Machine Learning, Long Beach, CA, 2017.
[17] M. Koyutürk, A. Grama, N. Ramakrishnan, Nonorthogonal decomposition of binary matrices for bounded-error data compression and analysis, ACM Trans. Math. Softw. 32 (2006) 33–69.
[18] D. Lao, Netflix movie recommendation, https://www.kaggle.com/laowingkin/netflix-movie-recommendation, 2019.
[19] D.D. Lee, H.S. Seung, Learning the parts of objects by non-negative matrix factorization, Nature 401 (1999) 788.
[20] H. Liu, J. Sun, J. Guan, J. Zheng, S. Zhou, Improving compound–protein interaction prediction by building up highly credible negative samples, Bioinformatics 31 (2015) i221–i229.
[21] H. Lu, J. Vaidya, V. Atluri, H. Shin, L. Jiang, Weighted rank-one binary matrix factorization, in: Proceedings of the 2011 SIAM International Conference on Data Mining, SIAM, 2011, pp. 283–294.
[22] F. Maxwell Harper, J. Konstan, The MovieLens datasets: history and context. ACM transactions on interactive intelligent systems (TiiS), https://doi.org/10.1145/2827872.

[23] R. Medellín, J. González Sern, B. Vargas-Govea, https://www.kaggle.com/uciml/restaurant-data-with-consumer-ratings. (Accessed 2018 November 30).

[24] P. Miettinen, T. Mielikäinen, A. Gionis, G. Das, H. Mannila, The discrete basis problem, IEEE Trans. Knowl. Data Eng. 20 (2008) 1348–1362.

[25] S.H. Mirisaee, E. Gaussier, A. Termier, Improved local search for binary matrix factorization, in: AAAI, 2015, pp. 1198–1204.

[26] Q. Qi, Y. Zhao, M. Li, R. Simon, Non-negative matrix factorization of gene expression profiles: a plug-in for brb-arraytools, Bioinformatics 25 (2009) 545–547.

[27] B.-H. Shen, S. Ji, J. Ye, Mining discrete patterns via binary matrix factorization, in: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2009, pp. 757–766.

[28] J. Tanner, K. Wei, Low rank matrix completion by alternating steepest descent methods, Appl. Comput. Harmon. Anal. 40 (2016) 417–429.

[29] R.K. Vinayak, S. Oymak, B. Hassibi, Graph clustering with missing data: convex algorithms and analysis, in: Advances in Neural Information Processing Systems, 2014, pp. 2996–3004.

[30] Y. Wang, A. Singh, Provably correct algorithms for matrix column subset selection with selectively sampled data, J. Mach. Learn. Res. 18 (2017) 5699–5740.

[31] J. Xu, R. Wu, K. Zhu, B. Hajek, R. Srikant, L. Ying, Jointly Clustering Rows and Columns of Binary Matrices: Algorithms and Trade-Offs, ACM SIGMETRICS Performance Evaluation Review, vol. 42, ACM, 2014, pp. 29–41.

[32] M. Xu, R. Jin, Z.-H. Zhou, Cur algorithm for partially observed matrices, in: International Conference on Machine Learning, 2015, pp. 1412–1421.

[33] Y. Xu, W. Yin, Z. Wen, Y. Zhang, An alternating direction algorithm for matrix completion with nonnegative factors, Front. Math. China 7 (2012) 365–384.

[34] P. Yadava, Boolean matrix factorization with missing values, master's thesis, Universitat des Saarlandes, 2012.

[35] Z.-Y. Zhang, T. Li, C. Ding, X.-W. Ren, X.-S. Zhang, Binary matrix factorization for analyzing gene expression data, Data Min. Knowl. Discov. 20 (2010) 28.