Reconstructing Graphs from Neighborhood Data

DÓRA ERDŐS, Boston University
RAINER GEMULLA, Max Planck Institut für Informatik
EVIMARIA TERZI, Boston University

Consider a social network and suppose that we are only given the number of common friends between each pair of users. Can we reconstruct the underlying network? Similarly, consider a set of documents and the words that appear in them. If we only know the number of common words for every pair of documents, as well as the number of common documents for every pair of words, can we infer which words appear in which documents? In this article, we develop a general methodology for answering questions like these.

We formalize these questions in what we call the RECONSTRUCT problem: given information about the common neighbors of nodes in a network, our goal is to reconstruct the hidden binary matrix that indicates the presence or absence of relationships between individual nodes. In fact, we propose two different variants of this problem: one where the number of connections of every node (i.e., the degree of every node) is known and a second one where it is unknown. We call these variants the *degree-aware* and the *degree-oblivious* versions of the RECONSTRUCT problem, respectively.

Our algorithms for both variants exploit the properties of the singular value decomposition of the hidden binary matrix. More specifically, we show that using the available neighborhood information, we can reconstruct the hidden matrix by finding the components of its singular value decomposition and then combining them appropriately. Our extensive experimental study suggests that our methods are able to reconstruct binary matrices of different characteristics with up to 100% accuracy.

Categories and Subject Descriptors: H.2.8 [**Database Management**]: Database Applications—*Data mining*

General Terms: Algorithms, Theory, Experimentation

Additional Key Words and Phrases: Bipartite graph reconstruction, adjacency matrix, singular value decomposition

ACM Reference Format:
Dóra Erdős, Rainer Gemulla, and Evimaria Terzi. 2014. Reconstructing graphs from neighborhood data. ACM Trans. Knowl. Discov. Data 8, 4, Article 23 (August 2014), 22 pages.
DOI: http://dx.doi.org/10.1145/2641761

1. INTRODUCTION

The neighbors that are common between a pair of nodes of an undirected graph carry valuable information about the graph structure. For example, in the context of movie

A shorter version of this article has appeared in the proceedings of the IEEE International Conference on Data Mining (ICDM) 2012 [Erdös et al. 2012].
This research was supported in part by NSF grants CNS-1017529 and IIS-1218437 as well as gifts from Microsoft, Google and Yahoo!.
Authors' addresses: D. Erdos, Department of Computer Science, Boston University, 111 Cummington Mall, Boston MA 02215, USA; R. Gemulla, Max-Planck-Institut für Informatik, Databases and Information Systems Department, 66123 Saarbrücken, Germany. Author's current address: Chair of Practical Computer Science, Universität Mannheim, Parkring 39, D-68131 Mannheim, Germany; E. Terzi, Department of Computer Science, Boston University, 111 Cummington Mall, Boston MA 02215, USA.

DOI: http://dx.doi.org/10.1145/2641761

recommendations, we may say that two users are similar if they have watched the same or a largely overlapping set of movies. Likewise, two movies are similar if they have been watched by the same set of users. Such neighborhood information has been exploited successfully in collaborative-filtering algorithms [Baeza-Yates and Ribeiro-Neto 2011; Das et al. 2007; Koren 2010], which recommend movies to users based on the number (and ratings) of movies these users have in common with other users. As another example, the set of words that are shared between two documents is an indicator of the documents' topical similarity and is exploited by document clustering techniques [Baeza-Yates and Ribeiro-Neto 2011]. Finally, the set of the common friends or common interests between social network users carries valuable information about the strength and quality of their friendship [Goyal et al. 2010; Lattanzi and Sivakumar 2009; Zheleva et al. 2009].

Generally, the set of features (e.g., movies, groups, friends, or words) that are shared by two entities (e.g., users or documents) reveals valuable information for data-mining algorithms. Traditionally, this information is extracted directly from the available data, which explicitly states which features are associated with every entity (e.g., movies watched by users, words appearing in documents, friends or interests of a user).

In some cases, however, the original data contains sensitive private information that the dataset owner may not want to share. For example, Netflix may not want to share which customer watched which movie. Similarly, Facebook may be unwilling to share the friendship graph or the affiliation graph (i.e., the graph that contains information about the membership of users to groups). In such cases, the data owner can decide to reveal some *aggregate* form of the original data. Such aggregate should preserve enough valuable information for researchers and practitioners to test their data-mining methods and at the same time should hide the characteristics of individual entities.

In this article, we focus on a particular type of such aggregate information, which we call *neighborhood information*. The neighborhood information of a dataset reveals only the *number* of features shared by every pair of entities (and vice versa) but does not contain information about *which* features are shared. Given such neighborhood information, we try to answer the following question: "To what extent does the revelation of neighborhood information prevent an adversary from reconstructing the original dataset?" For example, in the domain of social networks, the question is whether or not we can identify the membership of users to groups given that we know only the number of common groups between every pair of users and the number of common users for every pair of groups.

We formalize this problem as a bipartite graph reconstruction problem, which we refer to as RECONSTRUCT. Intuitively, RECONSTRUCT assumes a hidden binary dataset that associates entities to features. This dataset can be represented as a bipartite graph, in which an edge between an entity p and a feature q indicates that q is observed in p. The input to the problem is encoded in the following *neighborhood information*: for every pair of entities p and p', we are given the number $\mathbf{L}(p, p')$ of features shared by the two entities. Similarly, for every pair of features q and q', we are given the number $\mathbf{R}(q, q')$ of entities associated with both features. Given $\mathbf{L}$ and $\mathbf{R}$, our goal is to reconstruct the hidden bipartite graph.

Here, we study two variants of RECONSTRUCT: the *degree-aware* (which we denote by RA) and the *degree-oblivious* (which we denote by RO) problems. In the first variant, we assume that the degree (i.e., the number of neighbors) of every node is given as part of the input. In the second variant, we assume that we only know the number of neighbors every node shares with every *other* node besides itself. That is, the degree of the nodes is unknown.

Apart from the new problem definitions, our main contribution lies in the design of heuristics that can effectively reconstruct the hidden dataset in both the RA and the

RO variants. The key observation that our algorithms exploit is that we can use the neighborhood information to (approximately) reconstruct parts of the singular value decomposition of the biadjacency matrix of the hidden bipartite graph. We investigate the utility of our methods on a variety of datasets from different application domains. We found that in most cases, the reconstruction error is low; in some cases, our algorithms are able to exactly reconstruct the hidden bipartite graph.

Roadmap: The rest of the article is organized as follows: We formally define the RECONSTRUCT problem and its two variants in Section 2. Our algorithms for the RA problem are presented in Section 4, while the algorithms for the RO problems are presented in Section 5. The results of our experimental study are presented in Section 6, and in Section 7 we give an in-depth discussion of the related work. Finally, we conclude the article in Section 8.

2. PROBLEM DEFINITION

The RECONSTRUCT problem can be expressed in terms of both bipartite graphs and binary matrices. Throughout our discussion, we will use both representations interchangeably. We assume that there exists a hidden bipartite graph $G = (P, Q, E)$, where P and Q constitute the sets of nodes in the left and the right partition, respectively. Set $n = |P|$ and $m = |Q|$. The edge set $E \subseteq P \times Q$ connects nodes from P with nodes in Q. Every bipartite graph can be represented by its *biadjacency matrix* $\mathbf{M}$. The biadjacency matrix is a binary $n \times m$ matrix with $M(p, q) = 1$ if and only if $(p, q) \in E$. For every node p from P (or Q), denote by $N(p)$ the set of neighbors of p in G.

In this article, we assume that G—and consequently $\mathbf{M}$—is hidden. Our goal is to construct $\mathbf{M}$ from aggregate information. As discussed previously, we focus on the case where the aggregate information consists of the number of common neighbors between all pairs of nodes. Formally, we assume that for each pair $(p, p') \in P \times P$, we are given $\mathbf{L}(p, p') = |N(p) \cap N(p')|$. Similarly, for each pair $(q, q') \in Q \times Q$, we are given $\mathbf{R}(q, q') = |N(q) \cap N(q')|$. We call $\mathbf{L}$ and $\mathbf{R}$ the *neighborhood matrices* of G.

Observe that the main diagonal of the neighborhood matrices contains the degree of each node. One may consider that revealing the degree of every node may reveal too much information about the node. For this reason, we consider two types of neighborhood matrices: (1) the *degree aware* and (2) the *degree oblivious*. The first ones are neighborhood matrices in which the main diagonal is known, while the latter are matrices in which the main diagonal is unknown.

Given $\mathbf{L}$ and $\mathbf{R}$, our goal is to find a *binary* matrix $\widehat{\mathbf{M}}$ that is as close to $\mathbf{M}$ as possible. Ideally, we aim to minimize the square of the Frobenius norm $F(\widehat{\mathbf{M}}, \mathbf{M}) = \|\widehat{\mathbf{M}} - \mathbf{M}\|_F^2$, where

$$\|\mathbf{X}\|_F^2 = \sum_{i=1}^{n} \sum_{j=1}^{m} \mathbf{X}(i, j)^2.$$

However, this objective $F(\widehat{\mathbf{M}}, \mathbf{M})$ cannot be computed since $\mathbf{M}$ is unknown (we are given only $\mathbf{L}$ and $\mathbf{R}$). We therefore quantify the quality of $\widehat{\mathbf{M}}$ with respect to $\mathbf{L}$ and $\mathbf{R}$. In more detail, our goal is to minimize the sum of

$$F_{\mathbf{L}}(\widehat{\mathbf{M}}) = \|\widehat{\mathbf{L}} - \mathbf{L}\|_F^2$$

and

$$F_{\mathbf{R}}(\widehat{\mathbf{M}}) = \|\widehat{\mathbf{R}} - \mathbf{R}\|_F^2,$$

where $\widehat{\mathbf{R}}$ and $\widehat{\mathbf{L}}$ denote the neighborhood information induced by $\widehat{\mathbf{M}}$.

PROBLEM 1 (RECONSTRUCT). *Given neighborhood matrices* $\mathbf{L}$ *and* $\mathbf{R}$*, find a binary matrix* $\widehat{\mathbf{M}}$ *that minimizes the sum* $F_{\mathbf{L}}(\widehat{\mathbf{M}}) + F_{\mathbf{R}}(\widehat{\mathbf{M}})$.

While our true objective is to find a binary matrix $\widehat{\mathbf{M}}$ that minimizes $F(\widehat{\mathbf{M}}, \mathbf{M})$, this is impossible to find as $\mathbf{M}$ is not known. However, since $\widehat{\mathbf{L}}$ and $\widehat{\mathbf{R}}$ are both related to $\widehat{\mathbf{M}}$ (see Observation 1 in Section 4 for the exact relationship), we believe that the optimal solution to problem 1 hints a good $\widehat{\mathbf{M}}$ as well. Specifically, it is easy to see that for any $\widehat{\mathbf{M}}$ where $F(\widehat{\mathbf{M}}, \mathbf{M})$ is minimal, $F_{\mathbf{L}}(\widehat{\mathbf{M}}) + F_{\mathbf{R}}(\widehat{\mathbf{M}})$ is also minimized with the $\widehat{\mathbf{L}}$ and $\widehat{\mathbf{R}}$ derived from $\widehat{\mathbf{M}}$. In Section 6, we show experimentally that in case $F_{\mathbf{L}}(\widehat{\mathbf{M}}) + F_{\mathbf{R}}(\widehat{\mathbf{M}})$ is low, then $F(\widehat{\mathbf{M}}, \mathbf{M})$ is too.

In case matrices $\mathbf{L}$ and $\mathbf{R}$ are degree oblivious, we modify the aforementioned definitions such that the main diagonals are not taken into account. Since the type of the neighborhood matrices will always be clear from the context, we abuse notation and write $F_{\mathbf{L}}(\widehat{\mathbf{M}})$ and $F_{\mathbf{R}}(\widehat{\mathbf{M}})$ for both degree-aware and degree-oblivious matrices. In what follows, we use RA (RO, respectively) to refer to the variant of the RECONSTRUCT problem where the input neighborhood matrices are degree aware (degree oblivious, respectively).

Discussion: The previous problem definitions as well as the algorithms presented in the next section also apply to general (nonbipartite) graphs: we simply use the graph's adjacency matrix instead of its biadjacency matrix (and redefine $\mathbf{L}$ and $\mathbf{R}$ appropriately). In fact, our algorithms are oblivious to the fact that the hidden matrix constitutes a biadjacency matrix of some graph; that is, they apply to any binary matrix. If $\mathbf{M}$ is symmetric, and thus $\mathbf{L} = \mathbf{R}$, then $\mathbf{M}$ can be thought of as the adjacency matrix of a general undirected graph. In case $\mathbf{M}$ is squared (thus of size $n \times n$) but asymmetric, then it can represent a general directed graph. Here $\mathbf{L}$ and $\mathbf{R}$ correspond to the neighborhood information of the outgoing (resp. incoming) links. In what follows, we focus on bipartite graphs for clarity of exposition.

3. BACKGROUND

Before describing our algorithms in detail, we provide some background on the *eigendecomposition* and the *singular value decomposition*. See Golub and Loan [1996] for an in-depth treatment of these decompositions.

The *eigendecomposition* of an arbitrary *symmetric* matrix $\mathbf{X}$ is a decomposition $\mathbf{X} = \mathbf{U}\Lambda\mathbf{U}^T$, where $\mathbf{U}$ is a unitary matrix having as columns the normalized eigenvectors of $\mathbf{X}$, and $\mathbf{\Lambda}$ is a diagonal matrix containing the corresponding eigenvalues of $\mathbf{X}$.

The *singular value decomposition (SVD)* of an arbitrary matrix $\mathbf{X}$ is a decomposition $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T$, where $\mathbf{U}$ ($\mathbf{V}$, respectively) is an orthonormal matrix with the left (right, respectively) singular vectors of $\mathbf{X}$ as its columns, and $\mathbf{\Sigma}$ is a diagonal matrix that contains the singular values of $\mathbf{X}$ in its main diagonal.

When matrix $\mathbf{X}$ is symmetric and positive semidefinite, then the left and the right singular vectors are equal. In this case, the eigendecomposition and the SVD decomposition coincide. The following fact is known about the SVD decompositions of a matrix $\mathbf{X}$ as well as matrices $\mathbf{X}\mathbf{X}^T$ and $\mathbf{X}^T\mathbf{X}$.

PROPOSITION 3.1. *If matrix* $\mathbf{X}$ *is an* $n \times m$ *matrix and its SVD is* $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$*, then the SVD decomposition of* $\mathbf{X}\mathbf{X}^T$ *is*

$$\mathbf{X}\mathbf{X}^T = \mathbf{U}\Sigma^2\mathbf{U}^T \tag{1}$$

and the SVD decomposition of $\mathbf{X}^T\mathbf{X}$ *is*

$$\mathbf{X}^T\mathbf{X} = \mathbf{V}\Sigma^2\mathbf{V}^T. \tag{2}$$

To see this, substitute $\mathbf{X}$ with $\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ in the left-hand side of Equation (1) to obtain

$$\begin{aligned}\mathbf{X}\mathbf{X}^T &= \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T(\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T)^T \\ &= \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\mathbf{V}\mathbf{\Sigma}\mathbf{U}^T = \mathbf{U}\mathbf{\Sigma}^2\mathbf{U}^T.\end{aligned}$$

The proof of Equation (2) is similar and omitted.

Moreover, it is known that a truncated SVD can give the best low-rank approximation of $\mathbf{X}$.

PROPOSITION 3.2 (ECKART AND YOUNG [1936]). *If $\mathbf{U}_k$ ($\mathbf{V}_k$, respectively) represents the left (right, respectively) singular vectors that correspond to the k singular values Σ_k of the largest magnitude, then matrix $\mathbf{X}_k = \mathbf{U}_k\Sigma_k\mathbf{V}_k^T$ is the best rank-k approximation of $\mathbf{X}$ in terms of the Frobenius norm. That is, $\mathbf{X}_k$ minimizes*

$$\|\mathbf{X} - \mathbf{X}_k\|_F^2 = \sum_{i=1}^{n}\sum_{j=1}^{m}\left(\mathbf{X}(i,j) - \mathbf{X}_k(i,j)\right)^2.$$

Observe that we can write $\mathbf{X}_k$ as the sum of k rank-1 matrices:

$$\mathbf{X}_k = \sum_{i=1}^{k}\mathbf{U}(:,i)\mathbf{\Sigma}(i,i)\mathbf{V}(:,i)^T. \tag{3}$$

Here $\mathbf{X}(:,i)$ denotes the ith column of $\mathbf{X}$. The decomposition of $\mathbf{X}_k$ into a sum of rank-1 matrices turns out to be useful for our estimation algorithms since it allow us to determine the elements of the singular value decomposition one component at a time.

4. SOLVING THE RA PROBLEM

The high-level idea of our algorithms is to compute $\widehat{\mathbf{M}}$ by reconstructing the components of the SVD of $\mathbf{M}$.

Recall that in the RA problem, we assume that the diagonal elements of the neighborhood matrices are equal to the degree of the nodes in the hidden bipartite graph. We refer to such neighborhood matrices by $\mathbf{L}_d$ and $\mathbf{R}_d$. The key to our approach for RA is the following observation.

OBSERVATION 1. *The matrices $\mathbf{L}_d$ and $\mathbf{R}_d$ are given by $\mathbf{L}_d = \mathbf{M}\mathbf{M}^T$ and $\mathbf{R}_d = \mathbf{M}^T\mathbf{M}$.*

This observation connects the hidden data matrix with the observed neighborhood matrices and allows us to use the singular value decomposition to devise an efficient heuristic algorithm.

Denote by $\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ the SVD of the *unknown* matrix $\mathbf{M}$. Combining Observation 1 with Proposition 3.1, we obtain $\mathbf{L}_d = \mathbf{U}\Lambda\mathbf{U}^T$ and $\mathbf{R}_d = \mathbf{V}\Lambda\mathbf{V}^T$. This means that the eigendecompositions of $\mathbf{L}_d$ and $\mathbf{R}_d$ provide the left and right singular vectors of $\mathbf{M}$. Additionally, we obtain $\mathbf{\Sigma} = \sqrt{\Lambda}$, where the square root is taken element-wise. Intuitively, our goal is to exploit this knowledge to reconstruct the unknown matrix $\mathbf{M}$. However, from here it is not immediately clear how to proceed, since we know the columns (eigenvectors) in $\mathbf{U}$ and $\mathbf{V}$ only up to sign. That is, while instead of the eigenvector $\mathbf{u} = \mathbf{U}(:,i)$ the vector $-\mathbf{u}$ could be used equivalently in the eigendecomposition of $\mathbf{L}$, this is not the case in the SVD of $\mathbf{M}$. Only one of the vectors $\mathbf{u}$ and $-\mathbf{u}$ yields the true SVD decomposition of $\mathbf{M}$, but we do not know which one. (The same can be said for the columns of $\mathbf{V}$.) To proceed, we make use of the observation that for a given eigenvector $\mathbf{u} = \mathbf{U}(:,i)$, we obtain the same product $\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ if we use $-\mathbf{u}$ or negate the corresponding singular value $\sigma_i = \mathbf{\Sigma}(i,i)$. (Again, the same holds for columns in $\mathbf{V}$.) For this reason, and for the rest of this article, we fix the columns of $\mathbf{U}$ and $\mathbf{V}$ to be whatever is output by the eigendecomposition algorithm applied to $\mathbf{L}$ and $\mathbf{R}$. Our task now is to assign positive

or negative signs to the singular values in $\mathbf{\Sigma}$ to obtain a suitable decomposition of $\mathbf{M}$. In this article, we are going to refer to this as *finding the sign of a singular value*.

More formally, set $\sigma_i = \mathbf{\Sigma}(i,i)$ and $\lambda_i = \mathbf{\Lambda}(i,i)$. We assume without loss of generality that the singular values are reported in decreasing order of magnitude, that is, $|\sigma_1| \geq |\sigma_2| \geq \cdots \geq |\sigma_n|$. By Proposition 3.1, we know that $\lambda_i = \sigma_i^2$ and thus $\sqrt{\lambda_i} = |\sigma_i|$. This means that σ_i can take two possible values: $-\sqrt{\lambda_i}$ and $\sqrt{\lambda_i}$. Let $\widehat{\mathbf{\Sigma}}$ be a diagonal matrix with values $\widehat{\sigma_1}, \ldots, \widehat{\sigma_n}$ in its main diagonal, such that $|\widehat{\sigma_i}| = |\sigma_i|$. Here each $\widehat{\sigma_i}$ is signed; that is, it is either positive or negative. We refer to $\widehat{\mathbf{\Sigma}}$ as a *sign assignment* of $\mathbf{\Sigma}$. Given a sign assignment $\widehat{\mathbf{\Sigma}}$, matrix $\mathcal{M} = \mathbf{U}\widehat{\mathbf{\Sigma}}\mathbf{V}^T$ constitutes an *estimate* of $\mathbf{M}$. Note that $\mathcal{M}$ may not be a binary matrix; we return to this issue later. Throughout the article, we use calligraphic capital letters to denote real-valued matrices related to $\mathbf{M}$ and thus avoid confusion between these matrices and the appropriate binary estimates $\widehat{\mathbf{M}}$.

Our algorithms aim to find the "best" sign assignment, that is, the one that produces the best estimate of $\mathbf{M}$. In order to do this, we need to address the following two questions: (1) How do we evaluate a given sign assignment? and (2) How can we find good sign assignments?

Evaluating sign assignments. Given $\mathbf{U}$ and $\mathbf{V}$ together with a sign assignment $\widehat{\mathbf{\Sigma}}$, we want to decide how well we can reconstruct $\mathbf{M}$. Ideally, we would like to evaluate $\widehat{\mathbf{\Sigma}}$ by computing $F(\mathcal{M}, \mathbf{M})$. Unfortunately, this approach is infeasible because $\mathbf{M}$ is unknown.

Alternatively, we could try to set $\mathcal{M} = \mathbf{U}\widehat{\mathbf{\Sigma}}\mathbf{V}^T$ and compute $F_{\mathbf{L}}(\mathcal{M}) + F_{\mathbf{R}}(\mathcal{M})$. However, this approach is also not helpful for two reasons. The first reason is that the quantities $F_{\mathbf{L}}(\mathcal{M})$ and $F_{\mathbf{R}}(\mathcal{M})$ do not depend on the sign assignment. To see this, observe that the elements of $\mathbf{\Sigma}$ get squared when we compute $\mathcal{L} = \mathcal{M}\mathcal{M}^T$ and $\mathcal{R} = \mathcal{M}^T\mathcal{M}$. Second, the matrix $\mathcal{M} = \mathbf{U}\widehat{\mathbf{\Sigma}}\mathbf{V}^T$ is not binary and our goal is to actually find a binary matrix.

In order to overcome these problems, we propose a way to evaluate the sign assignment $\widehat{\mathbf{\Sigma}}$, which utilizes the fact that $\mathbf{M}$ is a *binary* matrix: if $\mathcal{M}$ is a good estimate of $\mathbf{M}$, then it is close to $\mathbf{M}$ and—as a result—close to a *binary* matrix. For this, we define the *binary counterpart* $\text{bin}(\mathcal{M})$ of (any) matrix $\mathcal{M}$ to be the binary matrix closest to $\mathcal{M}$ in terms of the Frobenius norm. We denote the output of the bin( ) function by $\text{bin}(\mathcal{M}) = \widehat{\mathbf{M}}$. Then the values of $\widehat{\mathbf{M}}$ can be computed as

$$\widehat{\mathbf{M}}(i,j) = \begin{cases} 1 & \text{if } \mathcal{M}(i,j) > 0.5 \\ 0 & \text{else.} \end{cases}$$

In our case, we assume that matrix $\mathcal{M} = \mathbf{U}\widehat{\mathbf{\Sigma}}\mathbf{V}^T$ is a good estimate of $\mathbf{M}$ if it is close to its binary counterparts. Although this assumption is based on mere intuition, our experiments give strong evidence that the $\widehat{\mathbf{M}}$ minimizing $F(\mathcal{M}, \widehat{\mathbf{M}})$ leads to a low reconstruction error. While choosing 0.5 as a rounding threshold results in the binary $\widehat{\mathbf{M}}$ that is also the closest to $\mathcal{M}$, in our experiments (Section 6.3.2), we also show some results when different thresholds are used.

Observe that the binary matrix $\widehat{\mathbf{M}}$ output by the bin( ) function is unique. Based on this observation and although we do not make any formal claims, our hope is that there are few (or ideally just one) sign assignments that produce a binary matrix $\widehat{\mathbf{M}}$ that minimizes $F_L(\widehat{\mathbf{M}}) + F_R(\widehat{\mathbf{M}})$.

4.1. The RecSVD Algorithm

Using the intuitions we have developed in the previous paragraph, we describe here our algorithm for solving the RA problem. At a high level, our algorithm finds the best binary estimate of $\mathbf{M}$ by choosing the signs of the singular values greedily and then

using this sign assignment in order to estimate the final estimate $\widehat{\mathbf{M}}$. We discuss these two steps later.

ALGORITHM 1: `Greedy` algorithm to compute an optimalsign assignment $\widehat{\boldsymbol{\Sigma}}$ and construct $\mathcal{M}$ a nonbinary estimate of $\mathbf{M}$

Input: $\mathbf{U}_k$, $|\boldsymbol{\Sigma}_k|$, $\mathbf{V}_k$ and integer k
Output: $\widehat{\mathbf{M}}_k$
$\mathcal{M}_0 \leftarrow 0_{n\times m}$x
for $i = 1 \ldots k$ **do**
  $\mathcal{M}_i^+ = \mathcal{M}_{i-1} + \mathbf{U}_k(:,i)\sigma_i\mathbf{V}_k(:,i)^T$
  $\mathcal{M}_i^- = \mathcal{M}_{i-1} - \mathbf{U}_k(:,i)\sigma_i\mathbf{V}_k(:,i)^T$
  if $F(\mathcal{M}_i^+, \text{bin}(\mathcal{M}_i^+)) < F(\mathcal{M}_i^-, \text{bin}(\mathcal{M}_i^-))$ **then**
    $\mathcal{M}_i = \mathcal{M}_i^+$
  else
    $\mathcal{M}_i = \mathcal{M}_i^-$
  end if
end for
return $\mathcal{M}_k$

A `Greedy` sign assignment algorithm. Using matrices $\mathbf{U}$, $|\boldsymbol{\Sigma}|$, and $\mathbf{V}$ as inputs, the greedy sign assignment routine outputs a *nonbinary* matrix $\mathcal{M}$. The routine works iteratively. That is, it constructs $\mathcal{M}$ in k iterations, where k is a parameter that influences accuracy. In iteration i, the sign of σ_i is determined. In more detail, it computes matrices $\mathcal{M}_i^+$ and $\mathcal{M}_i^-$ by considering the positive and negative sign for σ_i, respectively. The signs of $\sigma_1, \ldots, \sigma_{i-1}$ are taken from previous iterations, and $\sigma_{i+1}, \ldots, \sigma_n$ are taken to be zero. Then the sign of the ith singular value is selected based on whether $\mathcal{M}_i^+$ or $\mathcal{M}_i^-$ is closer to its binary counterpart. The intuition behind this approach is that large singular values, which are processed first, have a significant impact on the estimate $\widehat{\mathbf{M}}$ so that we expect such greedy choices to lead to correct decisions. After k iterations have been completed, our algorithm produces a sign assignment $\widehat{\boldsymbol{\Sigma}_k}$ for the k singular values of the largest magnitude. Algorithm 1 gives pseudocode for this `Greedy` sign selection process. In the pseudocode, we demonstrate how $\mathcal{M}_i^+$ and $\mathcal{M}_i^-$ can be computed using $\mathcal{M}_{i-1}$; that is, there is no need for these matrices to be recomputed from scratch in every iteration. This observation leads to significant running-time improvements in practice.

The `RecSVD` algorithm. Here, we demonstrate how the `Greedy` routine we described earlier can be used to solve the RA problem. First, we compute the eigenvectors $\mathbf{U}$ and $\mathbf{V}$ as well as the eigenvalues $\boldsymbol{\Lambda}$ of both $\mathbf{L}$ and $\mathbf{R}$. Note that $\boldsymbol{\Lambda}$ is the same in both eigendecompositions. We then input matrices $\mathbf{U}$, $\sqrt{\boldsymbol{\Lambda}}$, and $\mathbf{V}$ to the `Greedy` algorithm to construct $\mathcal{M}$, which is subsequently rounded to a binary matrix $\widehat{\mathbf{M}}$. We refer to this complete algorithm as `RecSVD`. Given a value of k, `RecSVD` proceeds as follows:

—Compute the truncated SVD of $\mathbf{L}_d$ and $\mathbf{R}_d$:
  1: $\mathbf{L}_d \approx \mathbf{U}_k\boldsymbol{\Lambda}_k\mathbf{U}_k^T$
  2: $\mathbf{R}_d \approx \mathbf{V}_k\boldsymbol{\Lambda}_k\mathbf{V}_k^T$
—Run `Greedy`$(\mathbf{U}_k, \sqrt{\boldsymbol{\Lambda}_k}, \mathbf{V}_k, k)$ to obtain $\mathcal{M}_k$.
—Output $\widehat{\mathbf{M}}_k = \text{bin}(\mathcal{M}_k)$.

Observe that `RecSVD` does not use all of the left and right singular vectors obtained by the eigendecompositions of $\mathbf{L}_d$ and $\mathbf{R}_d$ but only the ones with the k eigenvalues of largest magnitude. For this reason, we compute only the truncated SVD—the k

largest singular values and the corresponding singular vectors—of $\mathbf{L}_d$ and $\mathbf{R}_d$ (see also Proposition 3.2), which significantly reduces computational costs.

Discussion. The impact of the number k of singular values used in the RECONSTRUCT problem is explored extensively in the experimental section (Section 6). In general, we expect the algorithms to work well, when the truncated rank-k SVD of $\mathbf{M}$ is a very close approximation of the original matrix $\mathbf{M}$. Clearly, if $\mathbf{M}$ is rank-k, then taking the singular values and singular vectors in our algorithm results in an $\widehat{\mathbf{M}}$ that is very close to $\mathbf{M}$. In practice, $\mathbf{M}$ is not low rank but usually has small *effective rank*. That is, there is usually a small number of singular values with a much higher magnitude than the rest. An example of such a matrix is a noisy block-diagonal matrix. We show some experimental results on such matrices in Section 6.2.

Running time of `RecSVD`. Assume w.l.o.g. that $n \geq m$. Since we only compute the k-largest-magnitude eigenvalues (and their corresponding eigenvectors) of $\mathbf{L}_d$ and $\mathbf{R}_d$, the running time of the eigendecomposition is $O(n^2k)$. The running time of the `Greedy` routine is $O(n^2k)$ as well so that the total time complexity of `RecSVD` is $O(n^2k)$.

Speeding up `RecSVD`. Since eigendecompositions are useful for many problems, there exists a vast majority of work devoted to speed up these computations (see, for example, the sampling-based approach in Drineas et al. [2006]). Such methods can be utilized—whenever needed—to speed up the first step of the `RecSVD` algorithm.

In what follows, we describe a technique to speed up the second step of `RecSVD`, that is, the `Greedy` algorithm given in Algorithm 1. Observe that the computations done in lines 3, 4, and 5 of Algorithm 1 require $O(n^2k)$ time. We can speed up this computation significantly by sampling the rows of $\mathbf{U}_k$ and $\mathbf{V}_k$. If we use a sample of c rows for some constant c, the running time is reduced to $O(c^2k)$. In order to choose the rows that affect the entries of the output matrix $\mathcal{M}$ the most, we sample row r with probability proportional to $\sum_{i=1}^{k} |\mathbf{U}_k(r,i) \cdot \mathbf{V}_k(r,i)|$.

5. SOLVING THE RO PROBLEM

In this section, we turn our attention to the RO problem. Recall that in this problem, our goal is to estimate $\mathbf{M}$ from degree-oblivious matrices $\mathbf{L}$ and $\mathbf{R}$. In this case, the degrees of nodes in P and Q are unknown; that is, the main diagonal elements of $\mathbf{L}$ and $\mathbf{R}$ are all zero.

The high-level idea for the solution of RO is to first reconstruct the main diagonals of $\mathbf{L}$ and $\mathbf{R}$, compute the corresponding singular value decomposition during the procedure, and then apply the `Greedy` heuristic to obtain the sign assignment. We present three algorithms; the first algorithm `Iterative` is efficient in reconstructing the main diagonals in case the nodes corresponding to the hidden $\mathbf{M}$ are such that they either have no neighbor in common or share most of their neighbors. The second, `LS`, can reconstruct the diagonals if $\mathbf{L}$ *and* $\mathbf{R}$ are both low rank. Finally, a hybrid approach, `Hybrid`, combines the benefits of both algorithms. In our experiments, we show that while `Iterative` and `LS` are efficient for certain types of underlying data, they can perform quite badly on others. On the other hand, the performance of `Hybrid` is almost identical to that of the performance of `RecSVD` on the RA problem on most of our datasets.

5.1. The `Iterative` Algorithm

First, we introduce a solution to the RO problem that is an iterative version of `RecSVD`. We call this new modified algorithm `Iterative`. `Iterative` starts with some initialization of the diagonal matrices $\widehat{\mathbf{D}}_P$ and $\widehat{\mathbf{D}}_Q$ corresponding to the estimated node degrees in P and Q, respectively—for example, $\mathbf{D}_P$ and $\mathbf{D}_Q$ could be initialized to zero. In every iteration,

we run RecSVD using input matrices $\widehat{\mathbf{L}}_d = \mathbf{L} + \widehat{\mathbf{D}}_P$ and $\widehat{\mathbf{R}}_d = \mathbf{R} + \widehat{\mathbf{D}}_Q$. Afterwards, we revise the node degrees by performing an educated guess of new values for $\widehat{\mathbf{D}}_P$ and $\widehat{\mathbf{D}}_Q$. These new values are computed using the output binary matrix $\widehat{\mathbf{M}}$ of RecSVD. That is, first we compute the updated versions of $\widehat{\mathbf{L}}_d = (\widehat{\mathbf{M}})(\widehat{\mathbf{M}})^T$ and $\widehat{\mathbf{R}}_d = (\widehat{\mathbf{M}})^T(\widehat{\mathbf{M}})$. Then new estimates of $\widehat{\mathbf{D}}_P$ and $\widehat{\mathbf{D}}_Q$ are obtained by the diagonal entries of $\widehat{\mathbf{L}}_d$ and $\widehat{\mathbf{R}}_d$, respectively.

Observe that in Iterative, we use only the main diagonals of $\widehat{\mathbf{L}}_d$ and $\widehat{\mathbf{R}}_d$ in order to update $\widehat{\mathbf{D}}_P$ and $\widehat{\mathbf{D}}_Q$. Hence, we can speed up computation by computing only the diagonals of $\widehat{\mathbf{L}}_d$ and $\widehat{\mathbf{R}}_d$.

Scheinerman and Tucker [2010] use a similar iterative approach to compute the eigenvalue decomposition of symmetric matrices with missing entries. They also show that convergence of the diagonals $\widehat{\mathbf{D}}_P$ and $\widehat{\mathbf{D}}_Q$ is not guaranteed. In practice, however, convergence is fast in most cases. In our experience, we found that 100 iterations sufficed to achieve convergence on all our datasets.

Running time of Iterative**.** Every iteration of Iterative performs a call to RecSVD. Under the assumption that $n > m$, we obtain $O(n^2k)$ time per iteration. Using the optimization mentioned earlier, the computation of $\widehat{\mathbf{D}}_P$ and $\widehat{\mathbf{D}}_Q$ takes at most $O(n^2)$ time. If Iterative is run for t iterations, then the total running time is $O(n^2kt)$.

Discussion. Our experimental investigation reveals that the reconstructive power of Iterative is dependent on the underlying dataset. The success of this algorithm (compared to RecSVD) depends mostly on how well it is able to reconstruct the main diagonals of $\mathbf{L}$ and $\mathbf{R}$. We find that Iterative performs best if nodes either have completely disjoint neighborhoods or share most of their neighbors. On the other hand, partially overlapping neighborhoods result in severe under- or overestimation of the diagonals. We believe that this is due to the fact that the SVD decomposition of a matrix is a best rank approximation; every singular value–singular vectors pair is contributing to the reconstruction of a squared area of the input matrix. Hence, it will optimize to approximate the majority of fields, and as a result, the estimate for a diagonal element will be proportional to the off-diagonal elements in its corresponding row and column rather than the true degree of that node.

5.2. The LS Algorithm

In our second algorithm, we use least-squares approximation to infer the missing main diagonal entries of the square matrices $\mathbf{L}$ and $\mathbf{R}$. We describe our algorithm LS for a general square rank-r matrix $\mathbf{A}$. Then, in our experiments, we apply this algorithm to both $\mathbf{L}$ and $\mathbf{R}$. An approach similar to this is used, for example, to infer missing values in traffic matrices by Bharti et al. [2010].

Let $\mathbf{A}$ be a square matrix, with unknown main diagonal elements. Assume $\text{rank}(\mathbf{A}) = r$. We estimate the elements of the main diagonal one by one. To compute a missing diagonal entry $\mathbf{A}(i, i)$, we apply the following heuristic. We first choose r independent rows, denoted by rows and r independent columns, denoted cols. These sets define the submatrix $\mathbf{Z} = \mathbf{A}(\texttt{rows}, \texttt{cols})$. We choose rows and cols to contain different indices, so that none of the missing diagonal elements are in $\mathbf{Z}$. We define the column vector $\mathbf{y}$ as $\mathbf{y} = \mathbf{A}(\texttt{rows}, i)$ and row vector $\mathbf{w}$ as $\mathbf{w} = \mathbf{A}(i, \texttt{cols})$. We then use least-squares approximation to learn the scaling factors $\boldsymbol{\beta}$ that satisfies $\mathbf{Z}\boldsymbol{\beta} = \mathbf{y}$. Given $\boldsymbol{\beta}$, we finally compute $\mathbf{A}(i, i)$ by Eq. (4):

$$\mathbf{A}(i, i) = \mathbf{w}\boldsymbol{\beta}. \tag{4}$$

A pictorial illustration of this is shown in Figure 1.

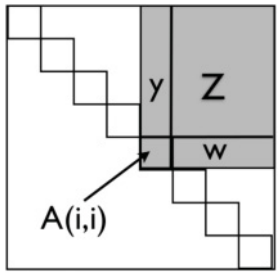


Fig. 1. The mechanics of the LS algorithm for reconstructing the main diagonal elements of $\mathbf{L}$ and $\mathbf{R}$.

In practice, if we apply this heuristic to $\mathbf{L}$ and $\mathbf{R}$, we are faced with the problem of not knowing the rank r of the neighborhood matrices. However, we can make use of the fact that all missing values are located in the main diagonal; we can choose `rows` and `cols` to contain up to $\frac{n}{2} - 1$ elements each, without containing any elements from the main diagonal. Though there is no theoretical guarantee, from our experiments we find that if the rank r of the matrix is less than $\frac{n}{2}$, then $\mathbf{Z}$ is very likely to have rank-r.

Running time of LS. For a given submatrix $\mathbf{Z}$, the least-square computation of $\boldsymbol{\beta}$ requires $O(n^2)$ time. Since there is a submatrix $\mathbf{Z}$ corresponding to every diagonal element $\mathbf{A}(i, i)$, the total running time is $O(n^3)$ assuming w.l.o.g. that the size of $\mathbf{A}$ is $n \times n$.

5.3. The Hybrid Algorithm

While both Iterative and LS algorithms are only efficient in underlying data with a certain structure, their combination leads to an algorithm that is consistently superior to both of them. We call this combination algorithm Hybrid. The key observation that led us to the design of Hybrid is the observation that the number of rows and columns in the biadjacency matrices $\mathbf{M}$ are not balanced. That is, if $\mathbf{M}$ is of size $n \times m$, then often either $n \gg m$ or $m \gg n$.

For the rest of the discussion (and without loss of generality), assume that $n \gg m$. Since $\mathbf{L}$ and $\mathbf{R}$ have the same rank, and this rank cannot be more than m, this means that $\mathbf{L}$ is a low-rank matrix. Given this, the Hybrid algorithm operates in two phases. In the first phase, we apply LS to $\mathbf{L}$ to reconstruct its main diagonal. Then, we use Iterative on $\mathbf{R}$. However, remember that in the latter algorithm, an SVD decomposition is computed in every iteration of the algorithm. Since we already reconstructed the full matrix $\mathbf{L}$, we can achieve higher accuracy by fixing $\boldsymbol{\Sigma}$ to be the square root of the eigenvalue matrix $\boldsymbol{\Lambda}$ of $\mathbf{L}$ and only update $\mathbf{V}$ in every iteration of the algorithm. Hence, the final algorithm Hybrid is the following:

—Compute $\mathbf{L}_d$ using LS.
—Compute $\mathbf{L}_d = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^T$, $\boldsymbol{\Sigma} = \sqrt{\boldsymbol{\Lambda}}$.
—Apply Iterative on $\mathbf{R}$, but fix $\boldsymbol{\Sigma}$.
—Run Greedy($\mathbf{U}_k$, $\sqrt{\boldsymbol{\Lambda}_k}$, $\mathbf{V}_k$, k) to obtain $\boldsymbol{\mathcal{M}}_k$.
—Output $\widehat{\mathbf{M}}_k = \text{bin}(\boldsymbol{\mathcal{M}}_k)$.

Running time of Hybrid. As the algorithm takes Iterative and LS as its subroutines, its running time depends on them. Hence, Hybrid takes $O(n^2kt + nr^3 \log r)$ time to construct $\mathbf{L}$ and $\mathbf{R}$. Then the second part of the algorithm has the same running time as Greedy that is $O(n^2k)$.

5.4. Comparison of Iterative, LS, and Hybrid

As the performance of the three algorithms introduced in this section depends strongly on the input data, we give a short analysis of our expectations and a preview of the experimental findings from Section 6 here.

As we mention in Section 5.1, Iterative is effective on input data where the nodes have a certain neighborhood structure. In our experiments, we show that the performance of Iterative varies on the real-life datasets with various different neighborhood structures.

LS (Section 5.2) works if both neighborhood matrices $\mathbf{L}$ and $\mathbf{R}$ are low rank. Furthermore, if this is not the case, then the algorithm will assign completely nonsense values to the main diagonals, only dependent on the particular implementation of the least-squares method but unrelated to the true hidden node degrees. As none of our real-life datasets are low rank in both $\mathbf{L}$ and $\mathbf{R}$, we decided not to show the experimental results separately for LS, since they would be similar to random guesses of $\widehat{\mathbf{M}}$.

Finally, we show the performance of Hybrid. Since most of our datasets have a biadjacency matrix $\mathbf{M}$ that is asymmetric in size (thus either $n > m$ or $n < m$), the LS algorithm applied to the neighborhood matrix that is low rank provides very good results. In case the appropriate neighborhood matrix is truly low rank (thus the difference between n and m is large), then LS gives a perfect estimate of $\mathbf{\Sigma}$, improving also the performance of Iterative in reconstructing the second neighborhood matrix. If $|n - m| > \frac{\min(n,m)}{2}$, then LS will provide correct estimates for some degrees and wrong estimates for others, depending on the random choice of $\mathbf{Z}$. In that case, the whole Hybrid algorithm that takes LS as a subroutine will suffer a bit in the performance. In this case, we will see that, depending on $|n - m|$, the performance of Iterative by itself or Hybrid is closer or farther away from each other.

6. EXPERIMENTS

In this section, we describe the results of an extensive experimental study; we evaluate our algorithms both with traditional measures—precision, recall, and the F-measure—and with some measures that we introduce that are specific to the RECONSTRUCT problem. We conduct our experiments on both synthetic and real-world datasets using a variety of data that represent the different structured data that may come up in real-life scenarios.

6.1. Experimental Setup

We conducted all experiments on a machine with Intel X5650 2.67GHz CPU and 12GB of memory. All algorithms were implemented in Matlab and the code is available at http://cs-people.bu.edu/edori/code.html.

Methodology. For all our experiments, the input to our algorithms consists of the neighborhood matrices that we compute as $\mathbf{L}_d = \mathbf{M} \cdot \mathbf{M}^T$ ($\mathbf{R}_d = \mathbf{M}^T \cdot \mathbf{M}$, respectively) for the RA problem. For RO, we set the main diagonal in $\mathbf{L}_d$ ($\mathbf{R}_d$) to zero to obtain $\mathbf{L}$ ($\mathbf{R}$, respectively). Although in all cases the underlying biadjacency matrix M is known, after generating the correct input, $\mathbf{M}$ is only used to evaluate the performance of our algorithms.

Evaluation metrics. To see how capable our algorithms are in reconstructing $\mathbf{M}$, we use traditional information retrieval statistics. That is, let $\widehat{\mathbf{M}}$ be the binary matrix that is output by one of our algorithms. Then we compare the zero and one entries in $\widehat{\mathbf{M}}$ and the original biadjacency matrix $\mathbf{M}$ to compute the precision, recall, and F-measure

assuming we use $\widehat{\mathbf{M}}$ to predict $\mathbf{M}$. We use the standard formulas

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{F-measure} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

with the notations TP, FP, and FN for true positive, false positive, and false negative, respectively.

We also report our results with regard to two further error metrics that show how well our algorithms are able to solve the RECONSTRUCT problem. The first metric is the *relative Frobenius error* (RFE) that incorporates the objective function of RECONSTRUCT and is given by

$$\text{RFE} = \frac{F_{\mathbf{L}}(\widehat{\mathbf{M}}) + F_{\mathbf{R}}(\widehat{\mathbf{M}})}{\|\mathbf{L}_d\|_F + \|\mathbf{R}_d\|_F}.$$

Observe that RFE $= 1$ when $\widehat{\mathbf{M}} = \mathbf{0}$ and RFE $= 0$ when $\widehat{\mathbf{M}} = \mathbf{M}$. Thus, RFE expresses the improvement over the all-zero solution to RECONSTRUCT with regard to the neighborhood information. Note that the sign assignment $\widehat{\Sigma}$ does affect the RFE since we use the binary counterpart $\widehat{\mathbf{M}}$ instead of using the real matrix $\mathcal{M}$ (cf. Section 4).

The second metric is the *relative absolute error* (RAE) measured with respect to the ground truth $\mathbf{M}$. It is given by

$$\text{RAE} = \frac{\|\widehat{\mathbf{M}} - \mathbf{M}\|_1}{\|\mathbf{M}\|_1},$$

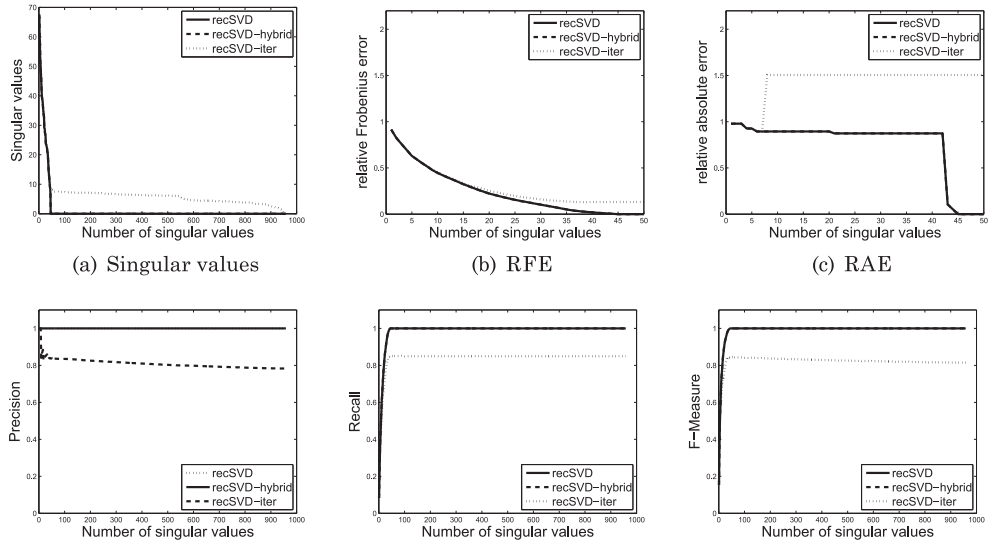
where $||\mathbf{X}||_1 = \sum_{ij} |\mathbf{X}(i, j)|$. The RAE measures the number of incorrect entries in estimate $\widehat{\mathbf{M}}$ relative to the number of nonzeros in $\mathbf{M}$. Thus, the all-zero solution obtains an RAE of 1.

6.2. Experiments with Synthetic Data

The first part of our experimental results is obtained through experiments on synthetically generated data. These experiments aim to illustrate the relationship between the performance of our methods and the structural characteristics of the underlying binary matrix $\mathbf{M}$.

6.2.1. Synthetic Dataset. As mentioned in Section 4, our algorithms explore the underlying low-rank data structure in the data matrix if existent. This is indicated by a low effective rank. For this reason, we conduct a set of experiments with a synthetic block-diagonal binary matrix. We first generate a set of rectangular blocks of all 1s with sizes chosen uniformly at random in between 1 and 100. The blocks are then arranged to form a block-diagonal matrix. This process generates a 0/1 matrix with only 1s appearing in each block. In order to obtain a full-rank matrix $\mathbf{M}$, we add some noise by randomly flipping 10% of the zeros to ones, and 10% of the ones to zeros. We refer to the resulting dataset as BLOCK; the particular instance used in our experiments contains 45 blocks and has size 2400×1000.

6.2.2. Comparison of the `RecSVD`*,* `Iterative`*, and* `Hybrid` *Algorithms on Synthetic Data.* First, it is worthwhile to look at the magnitude of the singular values of the input data



(a) Singular values (b) RFE (c) RAE

(d) Precision (e) Recall (f) F-measure

Fig. 2. Performance of the RecSVD, Iterative, and Hybrid algorithms on the BLOCK dataset; x-axis: number of singular values; y-axis: singular values (a), RFE (b), RAE (c), precision (d), recall (e), and F-measure (f).

(Figure 2(a)). As we mentioned in Section 4, the RecSVD algorithm can compute the magnitude of the singular values exactly since it is given by the neighborhood matrices $\mathbf{L}_d$ and $\mathbf{R}_d$. While $\mathbf{M}$ is full rank—it has 285 nonzero singular values—we can see from the solid line corresponding to RecSVD that the first 45 singular values are significantly larger than the others. This is not surprising as $\mathbf{M}$ consists of exactly 45 dense blocks of ones. The matrix is still full rank because of the 10% noise in the data; however, the corresponding singular values are insignificant. The line corresponding to Hybrid coincides with RecSVD, implying that we were able to perfectly estimate the dominant singular values of $\mathbf{M}$. On the other hand, while Iterative is perfect in reconstructing the first 35 singular values, then it fails to do so.

Keeping in mind that the effective rank of $\mathbf{M}$ is only 45, we can now direct our attention toward Figures 2(c) and 2(b) that contain the RFE and RAE achieved by the RecSVD, Iterative, and Hybrid algorithms. As we have established that only the first 45 singular values have any significance, for this dataset we show a zoomed-in version of these results, focusing on the first 45 values. As we can see, here both RecSVD and Hybrid algorithms show a steady decrease. One very obvious thing in this figure is that the dotted line corresponding to RecSVD in the RA problem coincides with Hybrid and hence is invisible. Thus, on this dataset, the Hybrid algorithm can reconstruct the original data matrix in the RO case as well as if the degrees of nodes were known. We can see that the performance of Iterative is better (decreasing) with regard to the RFE measure as RAE.

6.3. Experiments with Real Data

6.3.1. The Real Datasets. We perform experiments on several real-life datasets, which are examples of the applications mentioned in Section 1.

The MP3 dataset contains reviews for different models of mp3 players and was collected by Lappas et al. [2012]. The rows of the biadjacency matrix correspond to different models of mp3 players, while the right class corresponds to the features of mp3

players that are commented on in reviews. Cell (i, j) of this matrix is equal to 1 (0, respectively) if a the ith review comments (does not comment, respectively) on feature j of the mp3 player. The MP3 dataset consists of 711 reviews and 224 features and thus the corresponding biadjacency matrix is 711×224.

The CORA dataset consists of 2,708 scientific publications and a dictionary of 1,433 unique words. The dataset contains occurrences of words in the documents: we set $\mathbf{M}(d, w) = 1$ in the biadjacency matrix if document d contains word w. The dataset only contains words that have document frequency at least 10. The resulting bipartite graph contains about 50K edges.[1]

The FLICKR dataset contains information from the photo-sharing site `www.flickr.com`; it was provided by Zheleva and Getoor [2009]. Users of Flickr can form groups based on common interests. A user u is connected by an edge to group g if u is a member in g. We sample 2,000 users and 1,989 groups uniformly and at random from the original data. The resulting binary matrix that represents these associations contains about 2×10^6 entries with value equal to 1.

The YAHOO dataset encodes the membership of Yahoo! users into Yahoo! groups. The dataset is available to us through the Yahoo! Webscope project[2] The rows of the binary matrix correspond to users and columns correspond to groups. Entry (i, j) in this matrix is equal to 1 (0, respectively) if user i participates (does not participate, respectively) in the jth group. From the original Yahoo! Groups dataset, we select only a subset that consists of 15,874 users and 2,954 groups. That is, the size of the hidden biadjacency matrix has size $15{,}874 \times 2{,}954$ with 18K nonzero entries.

6.3.2. Effects of the Threshold in the Binary Rounding. Before we delve into the analysis of our algorithm, we conduct a set of initial experiments to see what effect the threshold t used to compute the binary counterpart of matrices has on our algorithm. As a reminder, the binary counterpart $\text{bin}(\mathcal{M})$ of a real matrix $\mathcal{M}$ is

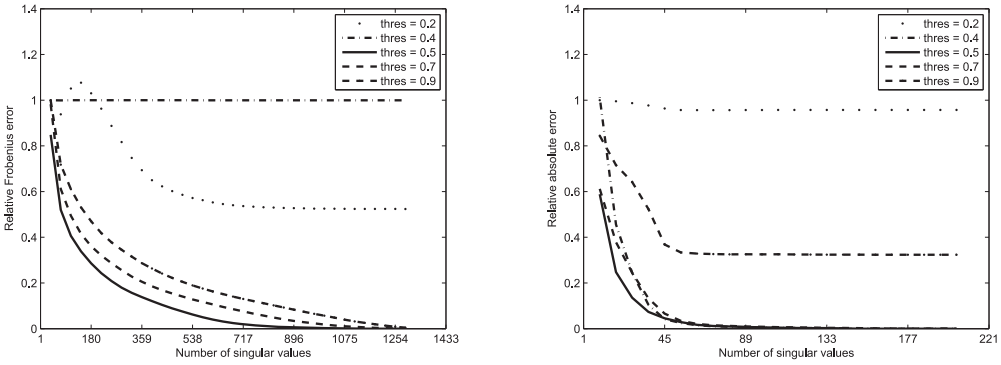
$$\widehat{\mathbf{M}}(i, j) = \begin{cases} 1 & \text{if } (i, j) > t \\ 0 & \text{else.} \end{cases}$$

The binary counterpart plays an important role in the `Greedy` part of our algorithm; we make greedy decisions to assign positive or negative signs to the singular values in the SVD decomposition based on the distance of a matrix and its binary counterpart.

As we have explained in Section 4, the threshold $t = 0.5$ results in binary counterpart matrices that are closest to their real counterparts. As we show in Figure 3, this choice will also result in the best performance of our algorithms. In Figure 3(a), we can see the RFE results of the `RecSVD` algorithm applied to the RA problem on the CORA dataset. The figure shows the RFE results (y-axis) corresponding to running `RecSVD` with different thresholds t for computing the binary counterpart as a function of the number of singular values used in the SVD decomposition (x-axis). As we can see, values $t < 0.5$ perform significantly worse than values $t > 0.5$. This is not surprising as the CORA dataset consists of a very sparse graph. A threshold that is smaller than 0.5 results in more ones in the binary matrix $\widehat{\mathbf{M}}$, which in turn accounts for more false-positive fields in the matrix. Similar trends can be observed for the MP3 data (Figure 3(b)), which is also a sparse dataset.

If we compare Figure 3(a) with Figure 3(b), we can see that the larger thresholds in case of the MP3 data perform very similarly to $t = 0.5$, while there is a more pronounced difference for the same thresholds in case of CORA. One explanation for this may be that the adjacency matrix of the MP3 dataset is much more structured; because the matrix is

[1]The CORA dataset is available at `http://www.cs.umd.edu/~sen/lbc-proj/LBC.html`.
[2]Available at `http://webscope.sandbox.yahoo.com`.



(a) RFE; CORA (b) RFE; MP3

Fig. 3. Relative Frobenius error for the RecSVD algorithm on the CORA (a) and MP3 (b) datasets. The different lines correspond to the different decision thresholds used when computing the binary counterpart of matrices in the Greedy part of RecSVD.; x-axis: number of singular values; y-axis: relative Frobenius error (RFE).

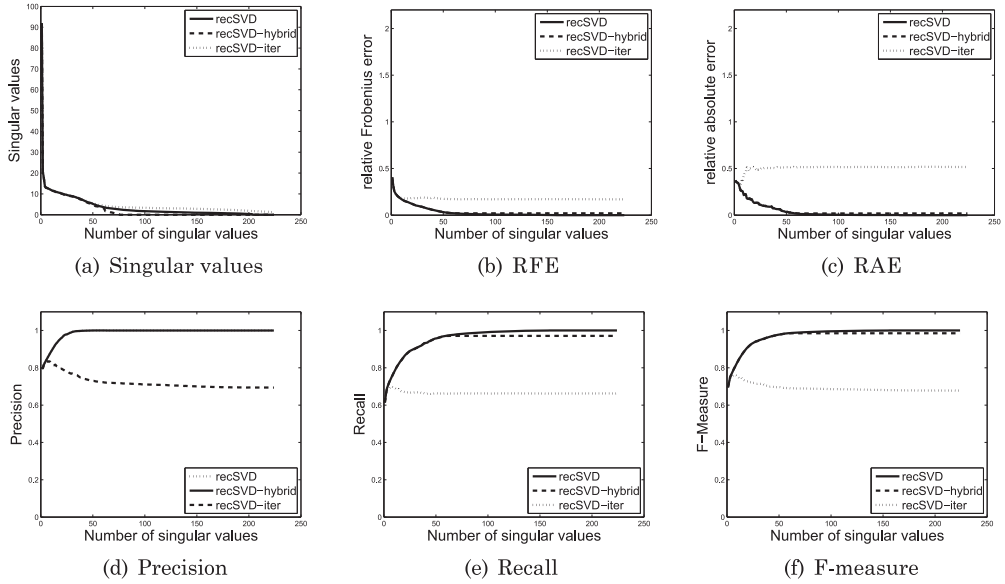
very asymmetrical in size but less noisy than the CORA data, there is a better one-to-one correspondence between nonzero values in $\mathbf{M}$ and the nonzero values generated by the singular vectors corresponding to a given singular value.

Here we only show the effect of the different threshold values t on the RecSVD algorithm for the RA problem with regard to RFE. However, we observed similar trends, namely, that $t = 0.5$ performs best and thresholds larger than $t = 0.5$ perform better than the smaller ones on sparse data, for all of our algorithms and evaluation measures across the datasets.

For the rest of the experiments, we use threshold $t = 0.5$.

6.3.3. Results on the MP3 *Dataset.* The first real-life dataset that we describe is the MP3 data. The size of the hidden biadjacency matrix is 711×224. Let us first look at the magnitude of the singular values of the adjacency matrix $\mathbf{M}$ predicted by our algorithms (Figure 4(a)). As we mentioned in Section 4, the RecSVD algorithm can compute the magnitude of the singular values exactly. Thus, the question Figure 4(a) addresses is how well Iterative and Hybrid approximate the singular values reported by RecSVD. Using the RecSVD as a baseline (solid line), we can infer that while $\mathbf{M}$ is full rank (thus $\text{rank}(\mathbf{M}) = 224$), the first 70 singular values are much more significant. We can also see that both Iterative and Hybrid—which solve the RO problem—are quite accurate in estimating the singular values. In fact, both these algorithms estimate the first 50 singular values perfectly. After this point, Iterative provides a slight overestimate and Hybrid an underestimate.

Figures 4(b) and 4(c) show the performance of our algorithms for both the RA and RO problems. Recall that for both measures, value 0 corresponds to the perfect reconstruction and a value of 1 is reported if $\widehat{\mathbf{M}}$ is estimated to be the $\mathbf{0}$ matrix. Also recall that both measures can take values significantly larger than 1—meaning that they can lead to significantly worse reconstructions. Taking this into consideration, we can observe that all our algorithms lead to good reconstructions. More specifically, RecSVD achieves values of almost 0 for both RFE and RAE using as few as 70 singular values. This shows the power of the binary rounding heuristic that we apply; we can use a low-rank approximation (the truncated SVD of $\mathbf{M}$ corresponding to the first 70 singular values) to actually get the full-rank matrix $\widehat{\mathbf{M}}$ that in this case is a perfect reconstruction of $\mathbf{M}$. In case of the Hybrid algorithm, we can see that it performs almost identically to



(a) Singular values (b) RFE (c) RAE

(d) Precision (e) Recall (f) F-measure

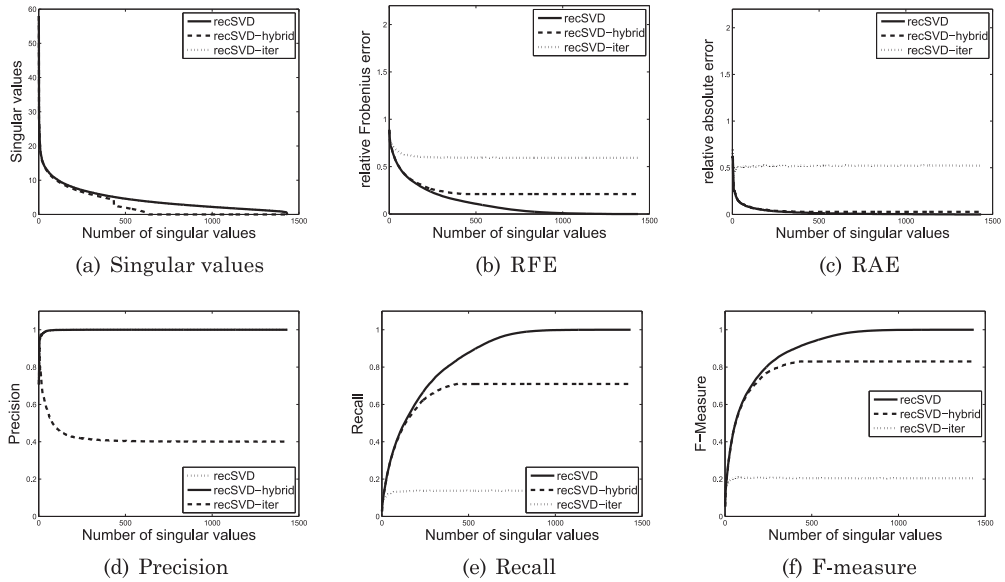
Fig. 4. MP3 dataset. Performance of the RecSVD, Iterative, and Hybrid algorithms as a function of the number of singular values; x-axis: number of singular values; y-axis: magnitude of the singular values (a), RFE (b), RAE (c), precision (d), recall (e), and F-measure (f).

RecSVD. For the first 70 singular values, RFE and RAE are in fact the same; after that Hybrid is very close to zero but never actually reaches that.

The MP3 dataset is interesting to our research especially, since it has the exact properties where Hybrid performs optimal; the left neighborhood matrix $\mathbf{L}$ is low rank, and hence the first part of the Hybrid approach is capable of reconstructing its main diagonal quite well. To show some specifics, the maximum degree in $\mathbf{L}_d$ is 60 and the average degree is 19.6. The average error that Hybrid achieves in reconstructing the degrees is 4.8 and the highest error is 15. Compared to Hybrid, the Iterative algorithm's performance is quite poor. However, we can see that for the first 20 singular values, it performs identically to the other two algorithms.

Finally, a joint observation of Figures 4(a), 4(b), and 4(c) demonstrates the following: whenever the estimates of the singular values were identical to the true values, the reconstruction potential of any of the two algorithms for RO is as strong as RecSVD for RA.

Some further intuition can be obtained by looking at the precision (Figure 4(d)), recall (Figure 4(e)), and F-measure (Figure 4(f)) achieved by the different algorithms. First, we can see that the precision and recall achieved by RecSVD become one quite soon; this overlaps with the 0 values in RFE and RAE. If we focus now on Iterative and Hybrid, it is first of all surprising to see that Iterative has the exact same precision as RecSVD, while Hybrid has a precision of around 0.7. If we look now at the recall, we can see a change in their relative performances. While Hybrid has a recall of about 0.98, the recall for Iterative is closer to 0.6. This explains the results we have seen for RFE and RAE for Iterative; Iterative predicts significantly more zeros in $\widehat{\mathbf{M}}$ than Hybrid. Since the one-valued entries predicted by Iterative are the subset of ones predicted by Hybrid, it naturally is capable of achieving better precision. However, we can see from the recall that as a result, it misses a lot of the true ones in the original data $\mathbf{M}$. If we look at the F-measure (Figure 4(f)), we can see that Hybrid performs overall much better than Iterative and almost identically to RecSVD.



(a) Singular values (b) RFE (c) RAE

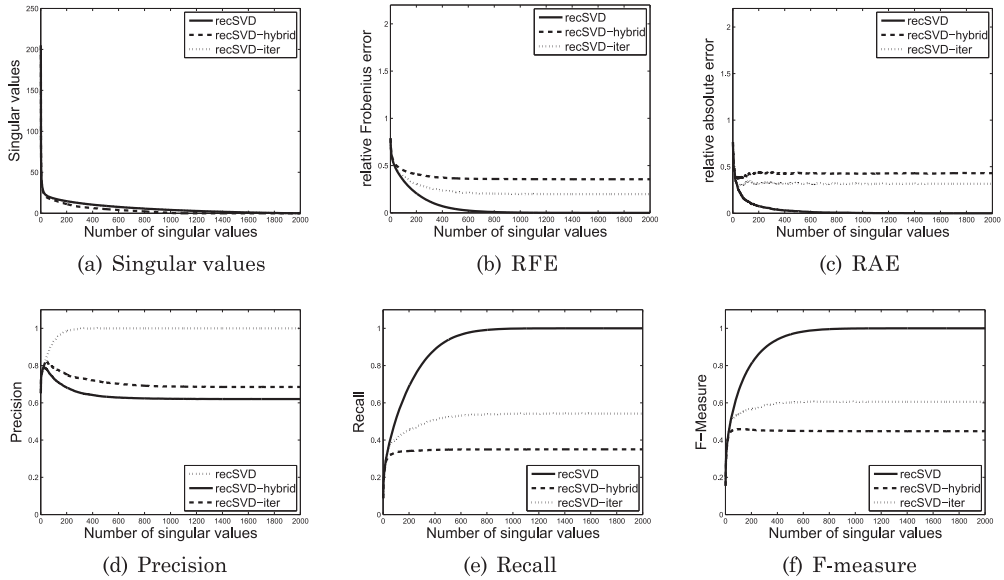
(d) Precision (e) Recall (f) F-measure

Fig. 5. CORA dataset. Performance of the RecSVD, Iterative, and Hybrid algorithms as a function of the number of singular values; x-axis: number of singular values; y-axis: magnitude of the singular values (a), RFE (b), RAE (c), precision (d), recall (e), and F-measure (f).

6.3.4. Results on the CORA *Dataset.* While the CORA dataset is still asymmetrical with respect to its number of rows and columns (2,708 vs. 1,433), it is not really true that the left neighborhood matrix $\mathbf{L}_d$ is truly low rank, as its rank is more than half its size. Further, if we look at the singular values of $\mathbf{M}$ (Figure 5(a)), we see that the decrease of the magnitude of singular values is gradual, and we cannot say that the effective rank of the matrix is much lower than its true rank. We can also see that in the RO problem, the singular values are underesimated in case of both algorithms.

If we look at the RFE results (Figure 5(b)), we see that RecSVD achieves 0 RFE after 1,000 singular values. Hybrid still performs better than Iterative. We can see that both these algorithms show decreasing RFE at first but then become constant. This can be attributed to the underestimated singular values; at some point the values in the estimated (real) matrix are only changing very little with the number of singular values taken into consideration; hence, the rounding process will assign the same zero or one value to every cell. We can see that the Hybrid performs closer to RecSVD with regard to RAE (Figure 5(c)). Precision (Figure 5(d)), recall (Figure 5(e)), and the F-measure (Figure 5(f)) tell us the same trends as in the case of the MP3 dataset, but the difference of the relative performance of the algorithms to each other is more pronounced.

6.3.5. Results on the FLICKR *Dataset.* The FLICKR dataset is different from the ones we considered in the previous paragraphs since the size of its left $\mathbf{L}$ and right $\mathbf{R}$ matrices are the same. Hence, both of them are full rank. As a result, the first part of the Hybrid algorithm—no matter whether it is applied to which side $\mathbf{L}$ or $\mathbf{R}$—will make almost random guesses about the main diagonal of the neighborhood matrix. As a result, we can see that the relative performance of Iterative and Hybrid changes when compared to the previous datasets. Overall, the experiments reported in Figure 6 demonstrate that across the various measures, Iterative is better. If we look at the precision (Figure 6(d)), we can observe that in this particular case the precision of RecSVD is below the other two. This is because both algorithms Iterative and Hybrid predict



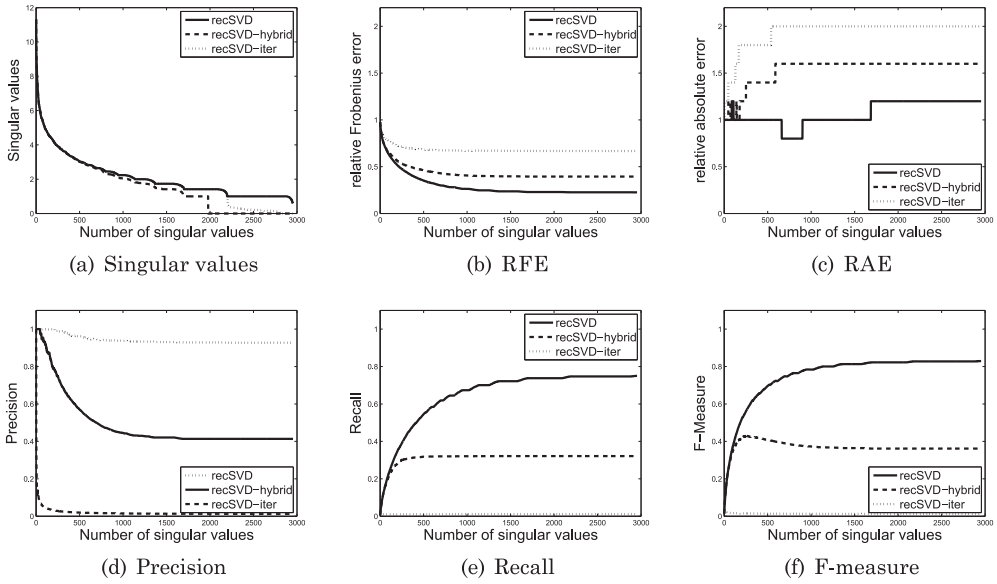
(a) Singular values (b) RFE (c) RAE

(d) Precision (e) Recall (f) F-measure

Fig. 6. FLICKR dataset. Performance of the RecSVD, Iterative, and Hybrid algorithms as a function of the number of singular values; x-axis: number of singular values; y-axis: magnitude of the singular values (a), RFE (b), RAE (c), precision (d), recall (e), and F-measure (f).

many zeros and hence achieve high precision. On the other hand, RecSVD significantly outperforms the other two on recall (Figure 6(e)).

6.3.6. Results on the YAHOO *Dataset.* Finally, we show results on the YAHOO dataset. This data can be best compared to the results on the MP3 dataset, as both biadjacency matrices $\mathbf{M}$ are asymmetrical in size. However, contrary to the MP3 dataset, the effective rank of the YAHOO data is equal to its rank. Hence, our low-rank reconstruction algorithms will perform less strongly than on the former dataset. The results on the YAHOO data are shown in Figure 7. Let us compare the results to those in Figure 4 corresponding to the MP3 data; first of all, both Iterative and Hybrid algorithms are able to reconstruct the larger singular values, but later both algorithms underestimate those (Figure 7(a)). It is important to note that in case of YAHOO, the magnitude of the singular values is gradually decreasing, and there is no effective rank that is much smaller than the true rank of $\mathbf{M}$. When looking at the performance results (Figures 7(b)–7(f)), we can see that the order of the algorithms is RecSVD being the best, Hybrid showing medium performance, and Iterative being the weakest. One exception is precision (Figure 7(d)), where Iterative is the best. This is again because Iterative predicts the most zeros among all algorithms and hence is capable of achieving very good results on the small number of ones in $\widehat{\mathbf{M}}$. Overall, we can see that while the relative performance of the algorithms compared to each other is similar in case of the YAHOO data as the MP3 data, the relative difference between the algorithms is more pronounced. Further, we can see that overall, all of the algorithms achieve somewhat worse results than on the MP3 data. We believe that this happens because the data is much more noisy in case of the YAHOO data. The difference between the two sizes of $\mathbf{M}$, $n = 15.8K$ and $m = 3K$, is large, and hence there is much more variety in the possible group memberships of YAHOO users than there was variety in the content of reviews in the MP3 data.



(a) Singular values (b) RFE (c) RAE

(d) Precision (e) Recall (f) F-measure

Fig. 7. YAHOO dataset. Performance of the `RecSVD`, `Iterative`, and `Hybrid` algorithms as a function of the number of singular values; x-axis: number of singular values; y-axis: magnitude of the singular values (a), RFE (b), RAE (c), precision (d), recall (e), and F-measure (f).

7. RELATED WORK

This work is an extended version of our previous work on the same topic [Erdös et al. 2012]. In this journal version, we have a more extensive discussion of both the RA and the RO problems. In addition to that, we propose two new algorithms for the RO problem, which clearly outperform the heuristics we had proposed in the conference version of the article. Finally, in addition to using more social network datasets than those we used in the original conference version, we also use datasets from different application domains (e.g., online product reviews).

To the best of our knowledge, the problem of reconstructing binary matrices from neighborhood information has—except for Erdös et al. [2012]—not been addressed before. However, our work is related to previous work on the analysis of real and binary matrices, as summarized here.

Matrix reconstruction. Existing work on matrix reconstruction focuses on the reconstruction of real-valued matrices from a few or noisy entries [Candès and Recht 2009; Keshavan et al. 2010]. Such reconstruction problems, also known as *matrix completion* problems, have received a lot of attention over the past years and existing studies have led to interesting algorithmic results. Although the goal of these methods is similar in spirit to our work in that they aim to reconstruct a hidden data matrix, they are tailored to real-valued matrices. In contrast, our work focuses on the reconstruction of binary matrices. More importantly, matrix completion methods use information about the values of a subset of the hidden matrix in order to reconstruct it. Such information is not available in our problem, in which only the per-node neighborhood information but no information about individual elements is provided.

Binary reconstruction. The problem of reconstructing binary matrices also arises in computer vision, for example. Here the goal is to reconstruct a noise-free image from

a noisy version of a black-and-white image. Existing methods for these problems use either combinatorial [Kolmogorov and Zabih 2004] or statistical inference techniques [Boykov et al. 1998]. These methods rely on the fact that a noisy version of the hidden matrix is known; the goal is to remove the noise from the observed pixels. Thus, these denoising techniques cannot be used to solve the RECONSTRUCT problem.

The problem of reconstructing 0/1 matrices has also been studied recently in data mining. For example, Vuokko and Terzi [2010] considered the problem where a randomized version of the data is revealed and the goal is to reconstruct the original data as accurately as possible. External knowledge about the relationships between the rows and the columns of the noisy observed matrix is also considered by the reconstruction algorithm. The method they proposed strongly relies on the fact that the observed matrix is a noisy version of the original one and that the amount of noise is known. Again, neighborhood information in the form considered in our work cannot be incorporated into the framework proposed in Vuokko and Terzi [2010].

Matrix decomposition. The analysis of binary data using matrix decompositions has been extensively studied [Miettinen 2008; Miettinen 2010; Miettinen and Vreeken 2011]. Although these methods do focus on 0/1 matrices, their goal is to find a low-rank representation of a known input matrix. In some sense, we try to achieve the opposite: we want to extract a hidden 0/1 matrix using some knowledge of its low-rank decompositions, as encoded in the neighborhood data.

Database security. Database privacy questions—in particular the possibility to reconstruct data from seemingly secure, anonymized information—have already been considered in the 1970s [Chin 1978]. The question here is what information can be inferred about the data if answers to a small set of statistical questions are known. More recent work by Mielikäinen [2004] is concerned with the inverse frequent itemset problem, which aims to infer the contents of a transaction database given an anonymized version of that database and true frequent itemset data.

Prediction of drug–protein interaction. A problem that at a high level resembles the RECONSTRUCT problem appears also in bioinformatics, where the goal is to predict drug–protein interactions. In that problem, the input consists of a set of drugs D, a set of protein targets P, a protein similarity matrix $\mathbf{S}_P$, a drug similarity matrix $\mathbf{S}_D$, and the interaction graph $\mathbf{M}(D, P)$, which is a bipartite graph representing the known molecular interactions between the proteins and the drugs. Given these, the goal is to predict an updated version of $\mathbf{M}$, called $\mathbf{M}^*$, given that a new drug is added to P or a new target is added to D.

One can view the drug–protein interaction graph $\mathbf{M}(D, P)$ as the hidden bipartite graph of the RECONSTRUCT problem. Also, the similarity matrices $\mathbf{S}_D$ and $\mathbf{S}_P$ can be mapped to the neighborhood matrices $\mathbf{L}_d$ and $\mathbf{R}_d$. Despite these high-level mappings, there are key differences. For example, matrices $\mathbf{S}_D$ and $\mathbf{S}_P$ take real values, while $\mathbf{L}_d$ and $\mathbf{R}_d$ are by definition restricted to be integers—since they represent the size of overlapping neighborhoods between two nodes.

Additionally, the drug–protein prediction problem has a different objective from the RECONSTRUCT problem. While in RECONSTRUCT $\mathbf{M}$ is completely unknown, here most part of $\mathbf{M}$ is known and the goal is only to predict the additional rows and columns. Current approaches to solve the drug–protein interaction prediction problem use methods from bipartite graph learning [Bleakley and Yamanishi 2009; Raymond and Kashima 2010; Bleakley et al. 2007] or semisupervised learning [Xia et al. 2010; Yamanishi et al. 2008; van Laarhoven et al. 2011]; However, these methods are not applicable to the RECONSTRUCT problem.

8. CONCLUSIONS AND DISCUSSION

There are countless types of real-life data that can be described in terms of a bipartite graph. In many cases, the original data is sensitive and cannot be made public. Thus, data owners may consider publishing aggregate information instead. In this article, we assume that the data owner reveals a particular type of aggregate information, that is, neighborhood information. The neighborhood information consists of the number of common neighbors between every pair of nodes. Given such information, we asked the following simple question: can we reconstruct the underlying graph using only this neighborhood information?

In this article, we formalized this question by introducing the RECONSTRUCT problem. The goal of this problem is to reconstruct the biadjacency matrix of a bipartite graph, given as input neighborhood information. Depending on whether the neighborhood information contains the number of common neighbors between all pairs of nodes, including the degree of the node itself, or not, we also defined two variants of the RECONSTRUCT problem, namely, the RA and the RO problems, respectively.

For those problems, we developed algorithms for reconstructing the hidden adjacency matrix. Intuitively, our methods reconstruct the components of the matrix' singular value decomposition from the neighborhood information. These components are subsequently used to obtain a (binary) estimate of the hidden matrix.

We study two versions of the problem; in the RA problem, the degrees of the nodes are known, while in the RO case, the degrees are hidden. Our experiments suggest that in case of RA, the underlying matrix can be reconstructed with low error for all datasets. For RO, we develop three algorithms that—along with reconstructing the underlying matrix—aim at reconstructing the degrees of the nodes as well. We find that while the efficiency of our algorithms depends on the underlying hidden data, our `Hybrid` algorithm reconstructs the matrix with quite high accuracy for various datasets.

We believe that our work is an instance of a more general problem, in which aggregate characteristics of a dataset are revealed and the question is whether one can infer individual data points. Computational tools that address such types of questions can prove valuable to data owners, who can use them to quantify how much information is leaked by the revealed aggregates.

REFERENCES

Ricardo Baeza-Yates and Berthier Ribeiro-Neto. 2011. *Modern Information Retrieval*. ACM Press/Addison Wesley.

Vineet Bharti, Pankaj Kankar, Lokesh Setia, Gonca Gürsun, Anukool Lakhina, and Mark Crovella. 2010. Inferring invisible traffic. In *CoNEXT*. ACM, 22.

Kevin Bleakley, Grard Biau, and Jean-Philippe Vert. 2007. Supervised reconstruction of biological networks with local models. *ISMB/ECCB Supplement of Bioinformatics*, (2007), 57—65.

Kevin Bleakley and Yoshihiro Yamanishi. 2009. Supervised prediction of drug-target interactions using bipartite local models. *Bioinformatics* 25, 18 (2009), 2397–2403.

Yuri Boykov, Olga Veksler, and Ramin Zabih. Markov random fields with efficient approximations. In *CVPR IEEE Conference on Computer Vision and Pattern Recognition*.

Emmanuel J. Candès and Benjamin Recht. 2009. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics* 9, 6 (2009), 717–772.

Francis Y. Chin. 1978. Security in statistical databases for queries with small counts. *ACM Trans. Database Syst.* 3, 1 (1978), 92–104.

Abhinandan Das, Mayur Datar, Ashutosh Garg, and ShyamSundar Rajaram. 2007. Google news personalization: Scalable online collaborative filtering. In *WWW International Conference on World Wide Web*. ACM, 271–280.

Petros Drineas, Ravi Kannan, and Michael W. Mahoney. 2006. Fast monte carlo algorithms for matrices II: Computing a low-rank approximation to a matrix. *SIAM J. Comput.* 36, 1 (2006), 158–183.

C. Eckhart and G. Young. 1936. The approximation of one matrix by another of lower rank. *Psychometrika* 1, 3 (1936), 211–218. DOI:http://dx.doi.org/10.1007/BF02288367

Dóra Erdös, Rainer Gemulla, and Evimaria Terzi. 2012. Reconstructing graphs from neighborhood data. In *ICDM IEEE International Conference on Data Mining*. IEEE Computer Society, 231–240.

Gene H. Golub and Charles F. Van Loan. 1996. *Matrix Computations*. The John Hopkins University Press.

Amit Goyal, Francesco Bonchi, and Laks V. S. Lakshmanan. 2010. Learning influence probabilities in social networks. In *WSDM International Conference on Web Search and Web Data Mining*. ACM, 241–250.

Raghunandan H. Keshavan, Andrea Montanari, and Sewoong Oh. 2010. Matrix completion from a few entries. *IEEE Trans. Inf. Theory* 56, 6 (2010), 2980–2998.

Vladimir Kolmogorov and Ramin Zabih. 2004. What energy functions can be minimized via graph cuts? *IEEE Trans. Pattern Anal. Mach. Intell.* 26, 2 (2004), 147–159.

Yehuda Koren. 2010. Factor in the neighbors: Scalable and accurate collaborative filtering. *TKDD Trans. Knowl. Discov. Data* 4, 1 (2010).

Theodoros Lappas, Mark Crovella, and Evimaria Terzi. 2012. Selecting a characteristic set of reviews. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 832–840.

Silvio Lattanzi and D. Sivakumar. 2009. Affiliation networks. In *STOC ACM Symposium on Theory of Computing*. ACM, 427–434.

Taneli Mielikäinen. 2004. Privacy problems with anonymized transaction databases. In *Discovery Science*. Springer, 219–229.

Pauli Miettinen. 2008. The boolean column and column-row matrix decompositions, *Data Min. Knowl. Discov.* 17, 1, (2008), 39–56.

Pauli Miettinen. 2010. Sparse Boolean matrix factorizations. In *ICDM IEEE International Conference on Data Mining*. IEEE Computer Society, 935–940.

Pauli Miettinen and Jilles Vreeken. 2011. Model order selection for Boolean matrix factorization. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 51–59.

Rudy Raymond and Hisashi Kashima. 2010. Fast and scalable algorithms for semi-supervised link prediction on static and dynamic graphs. In *ECML/PKDD*. 131–147.

Edward Scheinerman and Kimberly Tucker. 2010. Modeling graphs using dot product representations. *Computational Statistics* 25, 1 (2010), 1–16.

Twan van Laarhoven, Sander B. Nabuurs, and Elena Marchiori. 2011. Gaussian interaction profile kernels for predicting drug-target interaction. *Bioinformatics*. 27, 21, (2011), 3036–3043.

Niko Vuokko and Evimaria Terzi. 2010. Reconstructing randomized social networks. In *SDM SIAM International Conference on Data Mining*. SIAM, 49–59.

Zheng Xia, Ling-Yun Wu, Xiaobo Zhou, and Stephen TC Wong. 2010. Semi-supervised drug-protein interaction prediction from heterogeneous biological spaces. *BMC Syst. Biol.* (2010), 4(Suppl 2):S6.

Yoshihiro Yamanishi, Michihiro Araki, Alex Gutteridge, Wataru Honda, and Minoru Kanehisa. 2008. Prediction of drug target interaction networks from the integration of chemical and genomic spaces. In *ISMB (Supplement of Bioinformatics)*. 232–240.

Elena Zheleva and Lise Getoor. 2009. To join or not to join: The illusion of privacy in social networks with mixed public and private user profiles. In *WWW International Conference on World Wide Web*. ACM, 531–540.

Elena Zheleva, Hossam Sharara, and Lise Getoor. 2009. Co-evolution of social and affiliation networks. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1007–1016.

Received May 2013; revised October 2013; accepted December 2013