Goal-Directed Inductive Matrix Completion

Si Si
Dept. of Computer Science
University of Texas at Austin
ssi@cs.utexas.edu

Kai-Yang Chiang
Dept. of Computer Science
University of Texas at Austin
kychiang@cs.utexas.edu

Cho-Jui Hsieh
Depts. of Statistics and
Computer Science
University of California, Davis
chohsieh@ucdavis.edu

Nikhil Rao
Technicolor R&I
nikhilrao86@gmail.com

Inderjit S. Dhillon
Dept. of Computer Science
University of Texas at Austin
inderjit@cs.utexas.edu

ABSTRACT

Matrix completion (MC) with additional information has found wide applicability in several machine learning applications. Among algorithms for solving such problems, Inductive Matrix Completion(IMC) has drawn a considerable amount of attention, not only for its well established theoretical guarantees but also for its superior performance in various real-world applications. However, IMC based methods usually place very strong constraints on the quality of the features(side information) to ensure accurate recovery, which might not be met in practice. In this paper, we propose Goal-directed Inductive Matrix Completion(GIMC) to learn a nonlinear mapping of the features so that they satisfy the required properties. A key distinction between GIMC and IMC is that the feature mapping is learnt in a supervised manner, deviating from the traditional approach of unsupervised feature learning followed by model training. We establish the superiority of our method on several popular machine learning applications including multi-label learning, multi-class classification, and semi-supervised clustering.

1. INTRODUCTION

Matrix completion methods are widely used in several applications, ranging from collaborative filtering for recommender systems [8], to social network analysis [4] and clustering [25]. In these applications, a particular entry of the observed matrix is modeled as a linear interaction between factors corresponding to the row and column variables. For example, a rating provided by user i for movie j in a recommender system is modeled as $r_{ij} = \boldsymbol{w}_i^T \boldsymbol{h}_j$, where $\boldsymbol{w}_i, \boldsymbol{h}_j$ are low dimensional user and movie embeddings.

Modern applications typically involve large amounts of data (often in the millions or more), and several scalable algorithms have been proposed to solve matrix completion problems in such large data regimes [27, 21]. As a consequence of large-scale data acquisition methods, these applications involving large amounts of data also involve side information. Fast and scalable methods have been proposed in [30, 19] when the side information is in the form of pairwise relationships, such as product co-purchasing networks or social networks for recommender systems, or gene–gene interaction networks in computational biology.

In several applications of interest, one is not aware of the relational information in the form of graphs, but is given direct information in the form of features. These might correspond to demographic information (gender, occupation) for users and product information (genre, year of release) in a movie recommender system for example. When such features are provided, one can model an observation as a specific interaction between the features. For example in the recommender system case considered above, we would have $r_{ij} = \boldsymbol{x}_i^T Z \boldsymbol{y}_j$ where Z is the learnt model and $\boldsymbol{x}_i(\boldsymbol{y}_j)$ are the user (movie) features [5] [1]. These methods two advantages over traditional matrix completion applications:

- The target matrix to be estimated (Z above) can be significantly smaller than the partially observed data matrix, thus lowering the number of parameters to learn and speeding up computation.
- The presence of features allows the practitioner to generalize to previously unseen users and/or items, which is something standard matrix completion cannot do.

While such methods have been considered before in [24, 5], the authors in [2] showed that a significant drawback of feature-based matrix completion methods–Inductive Matrix Completion(IMC)– is that there need to be extremely strict constraints placed on the features to ensure accurate recovery of the underlying matrix. Specifically, the space spanned by the row (column) features must correspond to the row (column) space of the matrix to be factorized. In practice, such constraints are seldom met, which might result in poor performance of IMC. One way to solve this problem is by constructing mappings of the features so that the mapped features are more aligned to the subspaces in question. Moreover, the mappings should be constructed so that the methods are scalable, to be applicable to modern datasets. This brings us to the questions we wish to answer in this paper:

[1] We expound on this in more detail in the sequel

KDD '16, August 13 - 17, 2016, San Francisco, CA, USA

ISBN 978-1-4503-4232-2/16/08...$15.00

DOI: http://dx.doi.org/10.1145/2939672.2939809

1. Can a (possibly nonlinear) mapping be constructed for the features so that we can retain the gains provided by IMC given noisy features?

2. Can the resulting method be scaled to large datasets?

We answer both the above questions in the affirmative. To answer the first question, we aim to construct (nonlinear) embeddings of the row (or column) features so that they are better aligned to the task at hand. Specifically, we develop a minimization scheme called Goal-Directed Inductive Matrix Completion (GIMC) that alternates between learning the low-rank factors of the target matrix, and the aforementioned embeddings. A key contribution of this work is that the embeddings are learned in a supervised manner. This is a significant deviation from several other feature learning methodologies that learn the features in an unsupervised setting and use the learned features to solve the machine learning task.

To answer the second question, we resort to performing linear approximations of the nonlinear mappings that we aim to learn. Linear approximations of kernel-based methods have spawned large amounts of interest, either by using random Fourier features [17] or via Nyström approximations [23]. Such methods are attractive since they facilitate the use of highly scalable existing methods for learning with linear models. However, the dimension of these approximate kernel mappings could be very high, which induces a significant computational burden. We show that since we can learn the features based on the task at hand, we can use significantly fewer Fourier features or Nyström landmark points to achieve the same error as classical, unsupervised kernel approximation methods.

Viewed from a different perspective, our proposed Goal-Directed Inductive Matrix Completion framework could be viewed as a three-layer neural network. In this shallow neural network, the objective is a least squares loss and $cos(\cdot)$ and $sin(\cdot)$ are the activation functions. Interestingly, with such a shallow neural network, we achieve state-of-the-art performance in various matrix completion applications. Note that instead of using gradient descent in our algorithm to optimize the loss function, similar to neural networks, we could use back-propagation and SGD as an optimization scheme, and also use other activation functions such as the sigmoid function or hyperbolic tangent.

We consider three applications to test our proposed method: Multi-Label learning, Multi-Class classification, and Semi-supervised Clustering. In these three cases, we show that GIMC achieves state of the art results, while using far fewer features than what would be required to achieve the same performance using unsupervised feature learning methods. We verify that we achieve orders of magnitude speedups over existing methods.

The rest of the paper is organized as follows: in the next Section, we summarize related work. In Section 3 we detail the Inductive Matrix Factorization formulation and summarize known results. We propose our model in Section 4 and show how to apply our framework for solving three popular machine learning applications: Multi-Label learning, Multi-Class classification, and Semi-supervised Clustering in Section 5. We provide extensive empirical results on several real-world datasets in Section 6 before concluding our paper in Section 7.

2. RELATED WORK

We categorize related work into three categories, based on the core contribution of the corresponding works:

2.1 Scalable Matrix Factorization

Stochastic gradient descent(SGD) is widely used in matrix factorization due to its efficiency and ease of implementation. Various update schemes to parallelize SGD have been proposed: HogWild [21] uses lock free approach and DSGD [20] partitions the rating matrix into independent blocks to avoid conflict in the distributed system. [15] takes I/O cost and the CPU utility into consideration and proposed a fast and robust parallel SGD matrix factorization algorithm to deal with the situation when data cannot fit into memory. On a different note, [27][26] proposed to use coordinate descent instead of SGD to optimize the loss function, and the corresponding parallel solvers are shown to have superior performance in both multi-core and distributed settings.

2.2 Matrix Completion with Side Information

Matrix completion with side information has drawn much attention for improving the performance of traditional matrix completion. For instance, [19, 30] incorporates graph information as a regularization term into matrix completion framework, where the graph encodes pairwise relationships among variables. Along similar lines, [13] uses a social network among users to bias the recommender system. Instead of these pairwise relationships, in scenarios where one is provided with features for the rows and/or columns of the matrix, [5, 24, 10] proposed an Inductive Matrix Completion formulation. The method is "Inductive", in that it generalizes to previously unobserved data points, which is a drawback in traditional recommender systems. These methods have since been applied to predict gene-disease relationships [14] and semi-supervised clustering [25].

2.3 Non-Linear Kernel Mappings

To overcome the computational barrier of applying kernel methods to very large datasets, linear approximations have become popular, common among them being Nyström based methods [23] [29] [11] and random feature based methods [18] [12] [3, 7]. It has been shown that these approximate kernel mappings can speed up the training and prediction for kernel machines such as kernel SVM and Gaussian processes. To achieve a "good" approximation to the underlying kernel, these methods require the generation of many random features or landmark points. Thus, while such methods can in principle alleviate the computational costs associated with kernel machines, achieving a good approximation requires generation of many random features or landmark points which contributes to increased costs.

3. INDUCTIVE MATRIX COMPLETION

In this section, we first introduce traditional matrix completion and inductive matrix completion, and then formally set up the problem we are interested in solving. Consider a standard matrix completion setup, where we observe entries from an $n_x \times n_y$ matrix A. Let $\Omega : \quad |\Omega| \ll n_x n_y$ be the set of observed entries in A. Traditional matrix completion models A to be low rank, meaning that the row and column variables of A share a low dimensional latent space.

Standard matrix completion tries to recover the low-rank

matrix by solving one of the following optimization problems:

$$\min_{Z} \frac{1}{2}\|P_\Omega(A - Z)\|_F^2 + \lambda_z \|Z\|_*, \quad (1)$$

$$\min_{W,H} \frac{1}{2}\|P_\Omega(A - WH^T)\|_F^2 + \lambda_{WH}(\|W\|_F^2 + \|H\|_F^2), \quad (2)$$

where $W \in \mathbb{R}^{n_x \times k}$ and $H \in \mathbb{R}^{n_y \times k}$, and $P_\Omega(\cdot)$ is the projection operator that only retains those entries of the matrix that lie in the set Ω, It has been shown that these two problems are in fact equivalent when the chosen $k \geq \text{rank}(A)$ [22]. In addition, although the factorization objective (2) is non-convex, it turns out that the (local optimal) solution given by this non-convex form is usually comparable to the global optimum in (1). To solve (2), note that when either W or H is fixed, the above optimization becomes convex with respect to the other variable, so that one can solve (2) by alternating minimization. Various optimization techniques, for example, coordinate descent and stochastic gradient descent, have been proposed to solve (2) efficiently.

One challenge for traditional matrix completion is that it cannot directly use side information and apply it to predict new and unseen data, or to simplify computation. Inductive Matrix Completion [5] alleviates this issue. Specifically, suppose we are given a set of features $X \in \mathbb{R}^{n_x \times d_x}$ for the rows of A, and similarly for the columns, $Y \in \mathbb{R}^{n_y \times d_y}$. Each row of X (denoted as $\boldsymbol{x}_i$) and each row of Y (denoted as $\boldsymbol{y}_j$) are features for the i-th row and j-th column entity respectively. Then, IMC incorporates the feature information into matrix completion by solving the following problem:

$$\min_{Z} \frac{1}{2}\|P_\Omega(A - XZY^T)\|_F^2 + \lambda_z \|Z\|_*,$$
$$\min_{W,H} \frac{1}{2}\|P_\Omega(A - XWH^TY^T)\|_F^2 + \lambda_{WH}(\|W\|_F^2 + \|H\|_F^2), \quad (3)$$

where now $W \in \mathbb{R}^{d_x \times k}$ and $H \in \mathbb{R}^{d_y \times k}$.

Note that the number of unknown parameters are $k(d_x + d_y)$ which can be significantly smaller than $k(n_x + n_y)$. This has the potential to speed up computations by non-trivial amounts. Furthermore, besides recovering unseen entries in A, (3) can also be used to deal with cold start problem in recommendation systems. For example, for a movie recommendation problem, given a new user where only its features $\boldsymbol{x}_{\text{new}}$ are available, IMC could predict the score for the movies to be $\boldsymbol{x}_{\text{new}}^T ZY^T$. We refer the interested reader to [5] for details.

We consider a comparison between MC and IMC on a synthetic example. We generate the underlying matrix A as $XY^T + N$, where $X, Y \in \mathbb{R}^{200 \times 50}$ are two rank-50 random matrices with each entry drawn from Gaussian distribution $\mathcal{N}(0, 1)$, and $N \in \mathbb{R}^{200 \times 200}$ is the noise matrix with each entry drawn from $\mathcal{N}(0, 0.1)$. X and Y are provided as features to IMC, and rank k is set to be 20 for both MF and IMC. We vary the number of observed entries from 5% to 30% and recover the remaining entries of the target matrix. The approximately recovered matrix $\hat{A}$ will be WH^T in MF and XWH^TY^T in IMC. We then evaluate the recovered matrix $\hat{A}$ using the relative approximation error $\frac{\|\hat{A}-A\|_F}{\|A\|_F}$. As shown in Figure 1, IMC results in lower error than MC, suggesting that incorporating side information is useful for better recovery given a small perturbed observations.

While Figure 1 shows that IMC could outperform MC, the quality of the features plays a significant role in determining the accuracy of the recovered matrix. Indeed, the authors in [24] showed that to guarantee exact recovery in IMC, the features X, Y should be perfectly aligned with the row and column spaces of the target matrix A. Specifically, we require that

$$Col(A) \subseteq Col(X), \quad Row(A) \subseteq Col(Y), \quad (4)$$

where $Col(\cdot)$ and $Row(\cdot)$ correspond to the column and row space respectively. That is, to achieve good performance, the features X and Y should be strongly related to the underlying problem, which might not be satisfied in practice. This motivates us to ask if, given noisy features X and Y, one can learn an embedding of the features so that we can retain the gains provided by IMC given noisy features.

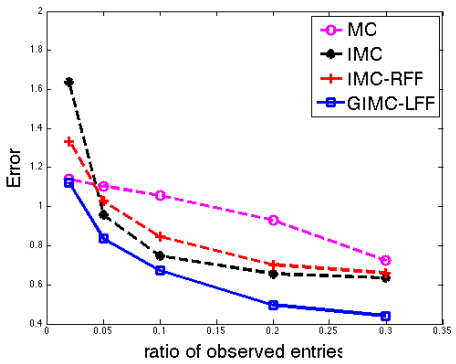


Figure 1: Toy example comparing MC, IMC, IMC-RFF, and GIMC-LFF. MC is the traditional model (Eq (2)); IMC is the model in Eq (3); IMC-RFF is a modification of IMC with random Fourier feature based side information as in Eq (5). GIMC-LFF learns the Fourier projections and the model simultaneously by solving Eq (9) side information is the random Fourier features from X and Y as in Eq (5); GIMC-LFF is to learn the projections of Fourier feature and model in IMC simultaneously. Using side information (IMC and IMC-RFF) improves performance, but not as significantly as learning both the projections and the model (GIMC-LFF)

4. PROPOSED FRAMEWORK: GIMC

In this section, we introduce our framework–Goal-directed Inductive Matrix Completion (GIMC). We first show how to employ nonlinear mappings into IMC, and then introduce the proposed GIMC model where we add supervision into nonlinear mappings throughout the learning process.

4.1 Nonlinear Feature Mapping for IMC

As discussed previously, performance of IMC may suffer if condition (4) is not satisfied. One way to address this issue is to generate features based on X and Y; that is, mapping X to $\Phi_X(X)$ and Y to $\Phi_Y(Y)$ via two mappings $\Phi_X : \mathbb{R}^{d_x} \to \mathbb{R}^{d'_x}$ and $\Phi_Y : \mathbb{R}^{d_y} \to \mathbb{R}^{d'_y}$. Notice that to overcome the violation of (4), such mapping has to be *nonlinear*, since for any linear mapping L, $Col(L(X)) \subseteq Col(X)$ and thus cannot help to relieve the violation of (4). By doing

so, the IMC problem can modified to be:

$$\min_{Z} \frac{1}{2}\|P_\Omega(A - \Phi_X(X)Z\Phi_Y(Y)^T)\|_F^2 + \lambda_z\|Z\|_*, \quad (5)$$

where $\Phi_X(X)$ and $\Phi_Y(Y)$ are new set of features generated by the non-linear mapping of X and Y respectively. For simplicity, here we again consider Z to be low rank, but in general, the term $\|Z\|_*$ could be further replaced by other regularizations for different applications. From now on, we will focus our discussion on the mapping of X, and use n, d to represent n_x, n_d. All subsequent discussions could be applied to Y as well with $n = n_y, d = d_y$.

Indeed, the specific mapping to be used can be varied, and the choices are many. A popular choice of such mappings are kernel mappings where we map $\boldsymbol{x}$ to $\varphi(\boldsymbol{x})$ which could be infinite dimensional. Recently, scalable alternatives to using exact kernels (either directly in the primal or via the Gram matrix in the dual) have been proposed where the exact mapping is replaced by an approximation. For instance, for shift-invariant kernels, based on Bochner's theorem[17], the feature mapping $\varphi(\cdot)$ can be written as (with high probability):

$$\boldsymbol{x}_i \to \varphi_U(\boldsymbol{x}_i) = \frac{1}{\sqrt{m}}[\cos(\boldsymbol{u}_1^T\boldsymbol{x}_i), \cdots, \cos(\boldsymbol{u}_m^T\boldsymbol{x}_i), \sin(\boldsymbol{u}_1^T\boldsymbol{x}_i), \cdots, \sin(\boldsymbol{u}_m^T\boldsymbol{x}_i)] \quad (6)$$

where $U = \{\boldsymbol{u}_1, \cdots, \boldsymbol{u}_m\}$ are the m projection directions sampled according to the distribution from the Fourier transform of the kernel function. For the RBF kernel ($k(\boldsymbol{x}_i, \boldsymbol{x}_j) = e^{-\gamma\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}$ with γ representing kernel width), the sampling distribution is a Gaussian distribution $p(\boldsymbol{u}) = N(0, 2\gamma I)$. Since there are infinite number of data points from that distribution, the feature mapping is infinite dimensional, and cannot be used in practice. [17] proposed to randomly sample a few projections from the distribution, and used the approximate feature mapping to speed up kernel machines.

Another popular kernel mapping is Nyström features that approximate the kernel matrix G by sampling $m \ll n$ landmark points $\{\boldsymbol{u}_r\}_{r=1}^m$, and forming two matrices $C \in \mathbb{R}^{n\times m}$ and $E \in \mathbb{R}^{m\times m}$ based on the kernel function. Here $C_{ir} = K(\boldsymbol{x}_i, \boldsymbol{u}_r)$ and $E_{ij} = K(\boldsymbol{u}_i, \boldsymbol{u}_j)$, and the kernel matrix can be approximated as

$$G \approx \bar{G} := CE^\dagger C^T, \quad (7)$$

where $E^\dagger$ denotes the pseudo-inverse of E. From the feature point of view, the Nyström method may be viewed as constructing features for $\boldsymbol{x}_i$; with m landmark points $\boldsymbol{u}_1, \cdots, \boldsymbol{u}_m$, the feature mapping is

$$\boldsymbol{x}_i \to \varphi_U(\boldsymbol{x}_i) = [k(\boldsymbol{x}_i, \boldsymbol{u}_1), k(\boldsymbol{x}_i, \boldsymbol{u}_2), \cdots, k(\boldsymbol{x}_i, \boldsymbol{u}_m)]M, \quad (8)$$

where $M = (E^\dagger)^{\frac{1}{2}}$ and $U = \{\boldsymbol{u}_1, \cdots, \boldsymbol{u}_m, M\}$.

However, when using the above feature mappings for solving IMC, as shown in the experiment, we need large number of projections or landmark points (large m), to achieve good performance. The memory requirement for storing these new set of features $\varphi_U(\boldsymbol{x}_i)$ for all $\boldsymbol{x}_i$ is $O(nm)$. This can be infeasible in many applications, for example, when $m > 10,000$ and n approaches 1 million. A natural question to ask is if we can reduce the number of samples needed to perform the approximation.

4.2 Goal-directed IMC

To overcome the issue with large number of nonlinear features, we propose Goal-directed Inductive Matrix Completion(GIMC), which aims to learn the feature mapping automatically to benefit the model. Instead of constructing a mapping $\varphi_U(\cdot)$ in an unsupervised setting which ignores the task at hand, we propose to learn the feature mappings and the model parameters simultaneously in a supervised manner. This can be attempted by solving the following joint optimization problem:

$$\min_{Z,U,V} \left\{ \lambda_z\|Z\|_* + \sum_{(i,j)\in\Omega} \ell(\varphi_U(\boldsymbol{x}_i)^T Z \varphi_V(\boldsymbol{y}_j), A_{ij}) \right\} := f(Z, U, V), \quad (9)$$

where ℓ is the loss function, Z represents the model, and the feature mapping of X and Y are $\varphi_U(\cdot), \varphi_V(\cdot)$ which are parameterized by U and V respectively. For simplicity, in this paper we focus on squared loss $\ell(b, a) = \frac{1}{2}(b - a)^2$, but in general we can extend the algorithm to any loss function. The main difference between problems (5) and (9) is that the former only considers the model parameters Z given a fixed feature mapping since U, V are predetermined, while the latter considers to learn Z and U, V simultaneously. As a result, we have the flexibility to optimize the kernel mapping according to the objective function that is defined by different machine learning applications with IMC.

Again, we can define the feature mapping $\varphi_U(\cdot)$ in (9) using various functions. In our first algorithm, Goal-directed IMC with Learned Fourier Features (GIMC-LFF) approach, we define the feature mapping according to (6), and the parameters $U = \{\boldsymbol{u}_r\}_{r=1}^m$ are the projection directions in Fourier Features. Compared to the original RFF that samples $\{\boldsymbol{u}_r\}_{r=1}^m$ randomly from a distribution, we can learn the projection directions $\{\boldsymbol{u}_r\}_{r=1}^m$ by minimizing the final objective function in IMC.

Our second proposed algorithm, Goal-directed IMC with Learned Nyström approximation (GIMC-LNYS), uses $\varphi_U(\cdot)$ defined in (8). In this case, the parameters $U = \{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m, M\}$, where each $\boldsymbol{u}_i$ is a landmark point, and $M \in \mathbb{R}^{m\times m}$ is a linear transformation. Although gradient descent updates discussed below can be applied to two set of variables ($\{\boldsymbol{u}_r\}_{r=1}^m$ and M), we observe no improvement by learning two sets of parameters, so in the following we assume $M = I$ in GIMC-LNYS. Thus, $U = \{\boldsymbol{u}_r\}_{r=1}^m$ for both GIMC-LNYS and GIMC-LFF, where each $\boldsymbol{u}_r \in \mathbb{R}^d$.

Problem (9) is non-convex and can be solved via alternating minimization, where we alternatively optimize the model parameters Z and feature mapping parameters U and V. The update of model Z varies through applications, and we will discuss how to update model parameters under specific applications shortly in Section 5. To update nonlinear mapping parameters $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m$, we use gradient descent and update the parameters as:

$$\boldsymbol{u}_r^{t+1} \leftarrow \boldsymbol{u}_r^t - \eta\nabla_{\boldsymbol{u}_r} f(Z, U, V), \ \forall r = 1, \ldots, m \quad (10)$$

where η is the step size and we adopt the Armijo-rule based step size selection to perform line search. Assume $B_{ij} = \varphi_U(\boldsymbol{x}_i)^T Z \varphi_V(\boldsymbol{y}_j)$ is the current prediction on (i, j) element,

the gradient for each $\boldsymbol{u}_r$ can be computed by chain rule:

$$\nabla_{\boldsymbol{u}_r} f(Z,U,V) = \sum_{(i,j)\in\Omega} \ell'(B_{ij}, A_{ij}) \frac{\partial \varphi_U(\boldsymbol{x}_i)}{\partial \boldsymbol{u}_r} \frac{\partial B_{ij}}{\partial \varphi_U(\boldsymbol{x}_i)}$$
$$= \sum_{(i,j)\in\Omega} \ell'(B_{ij}, A_{ij}) \frac{\partial \varphi_U(\boldsymbol{x}_i)}{\partial \boldsymbol{u}_r} Z\varphi_V(\boldsymbol{y}_j) \tag{11}$$

where $\frac{\partial \varphi_U(\boldsymbol{x}_i)}{\partial \boldsymbol{u}_r}$ is a $d \times m$ Jacobian matrix. For GIMC-LFF, each $\boldsymbol{u}_r$ only correlates with two features (see (6)), so the Jacobian matrix has only two nonzero columns; where for GIMC-LNYS each Jacobian matrix has only one nonzero column. The update of V could be conducted similarly.

The time complexity of computing (11) can be analyzed as follows. (1) Computing $\varphi_V(\boldsymbol{y}_j)$ for all j: Assume the computation of each element in $\varphi_V(\boldsymbol{y}_j)$ requires $O(d)$ time (one inner product for both GIMC-LNYS and GIMC-LFF), so constructing $\varphi_V(\boldsymbol{y}_j)$ for all j requires $O(dmn)$ time. This can be further reduced to $O(\text{nnz}(Y)m)$ if the original feature matrix Y is sparse. (2) Computing $\varphi_U(\boldsymbol{x}_i)$ for all i: similarly this requires $O(\text{nnz}(X)m)$ time. (3) Computing eq (11): Since only one or two columns of $\frac{\partial \varphi_U(\boldsymbol{x}_i)}{\partial \boldsymbol{u}_r}$ are nonzero, each term in eq (11) requires $O(m)$ time. In summary, the overall time complexity for evaluating $\nabla_{\boldsymbol{u}_r} f(Z,U,V)$ is

$$O\Big((\text{nnz}(X) + \text{nnz}(Y) + \text{nnz}(A))m\Big),$$

which is similar to the original matrix completion if m is small.

Details of the method are given in Algorithm 1. Since we employ alternating minimization scheme for solving (9), one can show that with the same number of landmark points or projection directions m, our method can achieve lower objective function values compared to the one achieved by using IMC with unsupervised Nyström (IMC-NYS) or random Fourier features(IMC-RFF). Specifically, let U^0 and V^0 be the initial set of mapping directions, Z^0 be the model trained using IMC-RFF or IMC-NYS, and U^t be the updated directions at iteration t of Algorithm 1, we will have

$$f(Z^t, U^t, V^t) \le f(Z^0, U^0, V^0),$$

which potentially implies a lower generalization error. In the toy example in Figure 1, we can clearly see that GIMC-LFF performs better than IMC-RFF, because we learn the feature and model together and thus construct a better set of non-linear features to benefit IMC. In our real-world experiments, we will also show that for a fixed number of features, our method outperforms IMC using both traditional Nyström and random Fourier feature mappings.

Algorithm 1: Goal-Directed Inductive Matrix Factorization (GIMC)

Input : Partially observed set $\{A_{ij} \mid (i,j) \in \Omega\}$, feature set $X = \{\boldsymbol{x}_i\}_{i=1}^{n_x}$ and $Y = \{\boldsymbol{y}_i\}_{i=1}^{n_y}$, number of landmark points or projection directions m_x, m_y for X and Y.

Output: The model Z, $U = \{\boldsymbol{u}_r\}_{r=1}^{m_x}$ and $V = \{\boldsymbol{v}_r\}_{r=1}^{m_y}$ which parameterize the goal-directed feature mapping for $X : \varphi_U(\cdot)$ and mapping for $Y : \varphi_V(\cdot)$ respectively.

[For GIMC-LFF]: Initialize $\{\boldsymbol{u}_r\}_{r=1}^{m_x}$, $\{\boldsymbol{v}_r\}_{r=1}^{m_y}$ by original random Fourier features

[For GIMC-LNYS]: Initialize $\{\boldsymbol{u}_r\}_{r=1}^{m_x}$, $\{\boldsymbol{v}_r\}_{r=1}^{m_y}$ by kmeans sampling or random sampling from training data

for $t = 1, \ldots,$ *maxiter* **do**

  Update Z by existing IMC with $\{\varphi_U(\boldsymbol{x}_i)\}_{i=1}^{n_x}$ and $\{\varphi_V(\boldsymbol{y}_i)\}_{i=1}^{n_y}$

  // Learn feature map for X

  Compute $\boldsymbol{g}_r \leftarrow \nabla_{\boldsymbol{u}_r} f(Z,U,V)$, $\forall r = 1, \ldots, m_x$

  Line search to find a step size η

  Update $\boldsymbol{u}_r \leftarrow \boldsymbol{u}_r - \eta \boldsymbol{g}_r$, $\forall r = 1, \ldots, m_x$

  // Learn feature map for Y

  Compute $\boldsymbol{g}_r \leftarrow \nabla_{\boldsymbol{v}_r} f(Z,U,V)$, $\forall r = 1, \ldots, m_y$

  Line search to find a step size η

  Update $\boldsymbol{v}_r \leftarrow \boldsymbol{v}_r - \eta \boldsymbol{g}_r$, $\forall r = 1, \ldots, m_y$

5. APPLICATIONS

We now state how to apply our proposed GIMC model to three machine learning applications: multi-label/multi-class learning problems, and semi-supervised clustering problem.

5.1 Multi-label and Multi-class Learning

Modern multi-label learning algorithms have to deal with problems with a very large number of samples, features and labels [2] For example in our experiments, the largest dataset we used has more than a hundred thousand samples, features, and labels.

Traditional multi-label learning approaches usually only consider linear models due to computational concerns, especially when the number of labels is huge (see [28, 16]). Using our goal-directed kernel approximation framework, we can not only learn the non-linear features, but also make computation efficient in both training and prediction phases.

Given training data $X = \{\boldsymbol{x}_i\}_{i=1}^n$, we use $A \in \mathbb{R}^{n \times L}$ to denote the 0/1 label matrix, where each row of A represents the L labels associated with $\boldsymbol{x}_i$. We can formulate a simple model that assumes no correlation among labels, and solve for each label independently:

$$\min_{\boldsymbol{w}_1 \ldots \boldsymbol{w}_L} \sum_{j=1}^{L} \Big(\frac{\lambda}{2} \|\boldsymbol{w}_j\|^2 + \sum_{i=1}^{n} \ell(\boldsymbol{w}_j^T \varphi_U(\boldsymbol{x}_i), A_{ij}) \Big),$$

where each $\boldsymbol{w}_j$ is the parameters for predicting whether the data has label j. This can be reduced to L binary classification problems, but this approach is very time consuming. For example, the Delicious dataset with 983 labels requires more than 1 day for training. Generally speaking, a method cannot scale to large n and large L if the time complexity grows with $O(nL)$.

We now show that we can apply GIMC-LFF and GIMC-LNYS to solve this problem. As ℓ is chosen to be the squared loss, the objective function can be rewritten as

$$\min_{W,U} \frac{\lambda}{2} \|W\|_F^2 + \frac{1}{2} \|A - \varphi_U(X)W\|_F^2,$$
$$\text{where} \quad \varphi_U(X) := [\varphi_U(\boldsymbol{x}_1) \ldots \varphi_U(\boldsymbol{x}_n)]^T, \tag{12}$$

and $W \in \mathbb{R}^{m \times L}$ are model parameters. We can see that (12) is a special case of (9), where $\varphi_V(Y) = I$ and Ω is

[2] called "extreme multi-label learning" http://research.microsoft.com/en-us/um/people/manik/events/xc15/

all the nL elements in A. [3] Therefore, we can solve it by the alternating minimization algorithm. Unfortunately, as analyzed in Section 4.2, directly computing (11) will lead to $O(|\Omega|m) = O(Lmn)$, which will not scale to problems with $L, n \geq 100,000$.

Interestingly, in the following we show that by carefully arranging the computation, the computational complexity can be reduced to $O(m^2(n+L)+m\|X\|_0+m\|A\|_0))$, which leads to a scalable algorithm that only depends on the number of nonzero elements in feature and label matrices.

When U is fixed, the subproblem with respect to W is a standard linear regression problem, which can be solved by a linear system solver: $W \leftarrow (\lambda\varphi_U(X)^T\varphi_U(X)+\lambda I)^{-1}\varphi_U(X)^T A$. This costs $O(m^3 + m\|A\|_0 + m^2L)$ time complexity.

When W is fixed, for GIMC-LFF the subproblem with respect to U can be written as

$$\nabla_{\boldsymbol{u}_r} f(\boldsymbol{w}, U) = \frac{1}{\sqrt{m}} \sum_{i=1}^{n}\sum_{j=1}^{L} (\boldsymbol{w}_j^T\varphi_U(\boldsymbol{x}_i) - A_{i,j})(W_{r,j}\sin(\boldsymbol{u}_r^T\boldsymbol{x}_i) - W_{r+m,j}\cos(\boldsymbol{u}_r^T\boldsymbol{x}_i))\boldsymbol{x}_i,$$

which can be written in a compact form as

$$\nabla_U f(\boldsymbol{w}, U) = \frac{1}{\sqrt{m}} X^T\big((\varphi_U(X)WW_{(2)}^T - AW_{(2)}^T)\circ\cos(XU)) - (\varphi_U(X)WW_{(1)}^T - AW_{(1)}^T)\circ\sin(XU))\big),$$

where $U \in \mathbb{R}^{d\times m}$, $W^T = [W_{(1)}^T\ W_{(2)}^T]$ and $\circ$ is the element-wise product. This can be computed in $O(m^2n + m\|X\|_0 + m\|A\|_0)$ time where $\|\cdot\|_0$ is number of nonzeroes in the matrix.

In the line search step, we use the following formulation to compute the objective function value,

$$\|A - \varphi_U(X)W\|_F^2 = \sum_{1\leq i,j\leq m}\big((WW^T)\circ(\varphi_U(X)^T\varphi_U(X))\big)_{ij} + \|A\|_F^2 - 2\sum_{1\leq i,j\leq m}(AW^T\circ\varphi_U(X))_{ij},$$

which only requires $O(m\|A\|_0 + nm^2 + Lm^2)$ time. The complexity will not grow as $O(nL)$, which means it can use large number of samples and labels when A is sparse. A similar update rule can be derived for GIMC-LNYS.

As a final remark, we can also use (12) for solving multi-class classification problem, where A is a $n \times L$ matrix with L to be the number of classes. $A_{ij} = 1$ when $\boldsymbol{x}_i$ belong to j-th class. The algorithm and analysis remain the same.

5.2 Semi-supervised Clustering

We now show that we can apply our GIMC framework for semi-supervised clustering as well. The semi-supervised clustering problem can be stated as follows: Suppose we are given n items and a feature matrix $X \in \mathbb{R}^{n\times d}$ where the i-th row $\boldsymbol{x}_i$ is the feature of i-th item, and a set of pairwise constraints $\mathcal{C} = \{S_{ij} \mid (i,j) \in \Omega\}$ where each constraint S_{ij} is in the following form:

$$S_{ij} = \begin{cases} 1, & \text{if item } i \text{ and } j \text{ are similar,} \\ 0, & \text{if item } i \text{ and } j \text{ are dissimilar.} \end{cases}$$

Then the goal is to find a clustering of items such that most similar items are within the same cluster. A state-of-the-art approach MCCC [25] models this problem using IMC framework. Specifically, let $S = \sum_{i=1}^{k}\boldsymbol{c}_i\boldsymbol{c}_i^T$ be the rank-k similarity matrix, where $\boldsymbol{c}_i$ is the indicator of the i-th cluster. Then the constraint set $\mathcal{C}$ could be thought of as partial observations of S. MCCC first tries to complete the low rank matrix S using IMC (3), and then uses its top-k singular vectors, which ideally will be the cluster indicators, to derive a partition.

Although it has been shown that MCCC outperforms traditional semi-supervised clustering algorithms [25], one issue is that it only considers linear mapping of features in matrix completion step, while the clustering may be revealed by some non-linear mapping of features. Thus, to overcome this problem, we propose to apply our GIMC framework to solve the semi-supervised clustering by learning both non-linear mapping of features and the underlying similarity S by solving the following optimization problem:

$$\min_{W,H,U,V} \frac{1}{2}\|P_\Omega\big(S - \varphi_U(X)WH^T\varphi_V(X)^T\big)\|_F^2 + \frac{\lambda}{2}(\|W\|_F^2 + \|H\|_F^2), \tag{13}$$

which is a special case of the GIMC framework (9) with $Y = X$ and $A = S$ [4]. To solve above optimization problem, we alternate between learning the nonlinear mapping (i.e. solve U, V with fixed W, H) and the model (i.e. solve W, H with fixed U, V). Once converged, we can derive a clustering by running k-means on top-k singular vectors of $\varphi_U(X)WH^T\varphi_V(X)^T$.

We now briefly discuss how to solve each subproblem efficiently using GIMC under semi-supervised clustering context. First, fixing U, V, solving W, H becomes a standard IMC problem, and one can use existing efficient algorithms (e.g. alternating minimization with conjugate gradient [28] as used in our experiment) to solve for W and H. The method addressed in [28] only requires $O((|\Omega| + nm)k)$ for each update of W or H.

Fixing W, H, we could solve U, V alternatively by gradient descent with line search. For example, let us first consider (13) to be GIMC-LFF where U, V are parameters for projections in Fourier Features, then the learning process of U is similar to the one in multilabel learning, except here we only consider the loss on a subset Ω. The gradient of each projection $\boldsymbol{u}_r$ could be written as:

$$\nabla_{\boldsymbol{u}_r} f(U, V, W, H) = \sum_{(i,j)\in\Omega}(\varphi(\boldsymbol{x}_i)^T\boldsymbol{q}_j - S_{ij})(-Q_{j,r}\sin(\boldsymbol{u}_r^T\boldsymbol{x}_i) + Q_{j,r+k}\cos(\boldsymbol{u}_r^T\boldsymbol{x}_i))\boldsymbol{x}_i, \tag{14}$$

where $Q = \varphi_V(X)HW^T$. The gradient of U could be computed in $O((|\Omega| + nd)m)$ as discussed in Section 4.2. In addition, for each line search iteration, evaluating objective function also takes $O((|\Omega|+nd)m)$ since only $|\Omega|$ inner products have to be computed. Similar update rule and time complexity analysis could also be derived if we consider Nyström features for U and V. Typically, $m, d \ll n, |\Omega| \ll n^2$, the overall time complexity for learning mappings grows at a rate much smaller than $O(n^2)$, especially if the available constraints are sparse. Using GIMC-LFF for semi-supervised

[3] Rigorously speaking, it is a special case for the "generalized" GIMC model where regularization of model parameter is replaced with $\|\cdot\|_F^2$.

[4] Here we consider the equivalent nonconvex form of IMC objective (see Section 3).

clustering is shown in Algorithm 2. We can derive similar algorithm using GIMC-LNYS.

Algorithm 2: Semi-supervised Clustering with GIMC-LFF

Input : feature matrix X, constraint set $\mathcal{C} = \{S_{ij} \mid (i,j) \in \Omega\}$, number of clusters k, regularization parameter λ, number of projections m.

Output: Clustering result π.

// Solve for GIMC-LFF model (13)

Initialize random projections $U, V \in \mathbb{R}^{d \times m}$, model parameter $W, H \in \mathbb{R}^{2m \times k}$.

for $t = 1, \ldots,$ *maxiter* **do**

  // Learning IMC model

  $[W, H] \leftarrow \min_{W,H} \frac{1}{2}\|P_\Omega\big(S - \varphi_U(X)WH^T\varphi_V(X)^T\big)\|_F^2 + \frac{\lambda}{2}(\|W\|_F^2 + \|H\|_F^2)$

  // Learning non-linear mapping

  Update U with gradient descent (using (14)).

  Update V with gradient descent (similar to (14)).

// Use left side of matrix to approximate top-k singular vectors

$T \leftarrow \text{svds}(\varphi_U(X)W, k)$

$\pi \leftarrow \text{kmeans}(T, k)$

6. EXPERIMENTS

We consider the three applications mentioned in the preceding section. We compare the following methods:

1. IMC-NYS: IMC with Nyström features. Here the landmark points in Nyström features are sampled uniformly at random from the dataset [23].

2. IMC-RFF: IMC with random Fourier features [18]. Random Fourier features are constructed based on the Fourier transform of the shift-invariant kernel function.

3. LEML[28]: state-of-the-art multi-label solver using IMC framework.

4. FastXML[16]: state-of-the-art multi-label solver using a random forest approach, where each tree is constructed by jointly optimizing both the ranking loss and tree structure.

5. SLEEC[1]: state-of-the-art multi-label solver which learns an ensemble of local distance preserving embeddings to preserve the pairwise distance between the neighbors' label vector.

6. k-means: traditional k-means clustering, which only considers data points as features.

7. MCCC[25]: a semi-supervised clustering method using IMC framework.

8. GIMC-LNYS: our proposed framework GIMC which learns Nyström features.

9. GIMC-LFF: our proposed framework GIMC which learns Fourier features.

Note that we compare with SLEEC, FastXML, and LEML for the multi-label learning task, and k-means and MCCC for semi-supervised clustering problem. SLEEC, FastXML, and LEML are highly optimized C++ implementation published along with the original papers. We use the parameters along with the released code for these three methods. We use Gaussian kernel as the basis kernel function used in Nyström feature and random Fourier features. The kernel width γ in Gaussian kernel and the regularization term λ in IMC are chosen by cross-validation. All experiments are conducted on a machine with an Intel Xeon X5440 2.83GHz CPU and 32G RAM.

6.1 Multi-label and Multi-class Learning

First, we show that our proposed algorithms can be applied to large-scale multi-label and multi-class classification problems. The datasets are listed in Table 1. The four multi-label datasets are from [1]. We evaluate the results by averaged precision at top 1, 3, and 5.

For multi-label learning problem, firstly we compare with IMC using Nyström features (IMC-NYS) and random Fourier features(IMC-RFF) in Figure 2. Here we vary m, i.e., the number of projections or landmark points in the feature mapping, which is directly related to the memory usage and prediction time of our method, and show the top-3 accuracy. It shows that GIMC-LFF and GIMC-LNYS are much better than IMC-NYS and IMC-RFF. This shows that with the same number of features, learning feature mapping and models together will benefit IMC. Furthermore, we compare with three state-of-the-art multi-label learning approaches: SLEEC [1], FastXML [16] and LEML [28] in the same figures. For these three algorithms, we compare with the best accuracy using the best parameters reported in their papers. Also, GIMC-LFF achieves better test accuracy compared to state-of-the-art multi-label solvers on all these three datasets.

The detailed comparison in terms of precision at $1, 3, 5$ are shown in Table 2. In addition, the efficiency of our algorithms is competitive to state-of-the-art algorithms, SLEEC and FastXML. On the NUS-WIDE dataset, using 500 features with 100 iterations, our GIMC-LFF achieves a 17.27% accuracy taking 6,482 seconds, while FastXML achieves a 16.32% using 17,774 seconds and SLEEC takes 68,232 seconds and achieves 17.67% accuracy.

We can also use our proposed methods for multi-class classification. We treat it as a special case of multi-label learning problem where the label matrix Y has only one 1 in each row. Here we test on one popular multi-class classification dataset: CIFAR-10.

CIFAR-10: We benchmark our method on the CIFAR-10[9], a popular multi-class classification dataset with 10 classes of natural images including 50,000 training samples and 10,000 testing samples. Each image is an 32× 32 RGB image. For each image, we extracted 12,288-dimensional features using CAFFE, a popular deep learning framework[6]. We then apply our proposed method GIMC-LFF and GIMC-LNYS on these new set of features for classification. The result is in Figure 2. In our experiment, linear SVMs with the new set of features achieves 81.73% accuracy, while our proposed methods achieve around 84% accuracy using 2000 features which is also much better than IMC with random Fourier features and Nyström features.

Table 1: Data set statistics for multi-label and multi-class classification problems.

Dataset	type	# training samples	# testing samples	# features	# labels
Bibtex	multi-label	4,880	2,515	1,836	159
Delicious	multi-label	12,920	3,185	500	983
NUS-WIDE	multi-label	161,789	107,859	1,134	1,000
Delicious-large	multi-label	196,606	100,095	782,585	205,443
CIFAR-10	multi-class	50,000	10,000	12,288	10

Table 2: Comparison between our proposed method GIMC with SLEEC, FastXML, and LEML for multi-label learning problem. Note that P1, P3, and P5 indicates Precision at top 1, top 3, and top 5 respectively. GIMC means GIMC-LFF for BibTex, Delicious, and NUS-WIDE datasets. For Delicious-large, we combine GIMC-LFF and GIMC-LNYS, that is, we concatenate the non-linear features learned from both GIMC-LFF and GIMC-LNYS, to make the final prediction. We can see that for the first three datasets, GIMC achieves the best performance comparing with other methods. On Delicious-large dataset, GIMC has similar accuracy with SLEEC, but takes much less time: 4,724 vs 25,289 seconds.

	Bibtex			Delicious			NUS-WIDE			Delicious-large		
	P1 (%)	P3 (%)	P5 (%)	P1 (%)	P3 (%)	P5 (%)	P1 (%)	P3 (%)	P5 (%)	P1 (%)	P3 (%)	P5 (%)
GIMC	**65.85**	**41.17**	**30.01**	**71.40**	**65.16**	**59.79**	**22.49**	**17.40**	**14.70**	46.13	40.32	38.15
SLEEC [1]	65.29	39.60	28.63	68.38	61.50	56.35	17.67	14.20	12.07	**47.03**	**41.67**	**38.88**
FastXML[16]	64.53	40.17	29.27	69.65	63.93	59.36	21.00	16.32	13.66	42.81	38.76	36.34
LEML[16]	62.53	38.40	28.21	65.66	61.15	56.08	20.76	16.00	13.11	40.30	37.76	36.66

6.2 Semi-Supervised Clustering

Since GIMC-LFF and GIMC-LNYS perform similarly well, we mainly compare our proposed GIMC-LFF for semi-supervised clustering with three popular clustering methods. We consider using simple k-means on data points' features (and ignore pairwise constraints) as the baseline, and also compare with state-of-the-art clustering method MCCC [25], and IMC-RFF which first generates random Fourier features, and then applies MCCC with these nonlinear features. We take $m = 500$ for both IMC-RFF and GIMC-LFF. Note that (1) MCCC is based on the IMC framework; (2) since MCCC has been shown to outperform many traditional methods (see [25] for details), we thus focus on comparing with MCCC to demonstrate the effectiveness of our model.

In this experiment, we consider three real-world datasets: Mushroom, Segment and Covtype [5] from classification benchmarks. Items with the same label are regarded as in the same cluster which is the general setting for semi-supervised clustering. We randomly sample $|\Omega|$ pairs of items and generate the pairwise constraints based on these pairs, with $|\Omega| = [1, 5, 10, 15, 20] \times n$, where n is the number of data points. For each choice of $|\Omega|$, we apply each method several times with different regularization parameters λ chosen from the set $\{0.1, 0.5, 1, 5, 10\}$, and take the parameter (and the corresponding clustering π) that achieves the smallest empirical clustering error on the validation set:

$$\frac{1}{|\Omega|}\Big(\sum_{(i,j)\in\Omega:\pi_i=\pi_j}\mathbf{1}(S_{ij}=0)+\sum_{(i,j)\in\Omega:\pi_i\neq\pi_j}\mathbf{1}(S_{ij}=1)\Big).$$

Then we use the parameter λ to train the model and generate the clustering π, and evaluate π using the clustering error rate to the ground-truth clustering defined by:

$$\frac{2}{n(n-1)}\Big(\sum_{(i,j):\pi_i^*=\pi_j^*}\mathbf{1}(\pi_i\neq\pi_j)+\sum_{(i,j):\pi_i^*\neq\pi_j^*}\mathbf{1}(\pi_i=\pi_j)\Big)$$

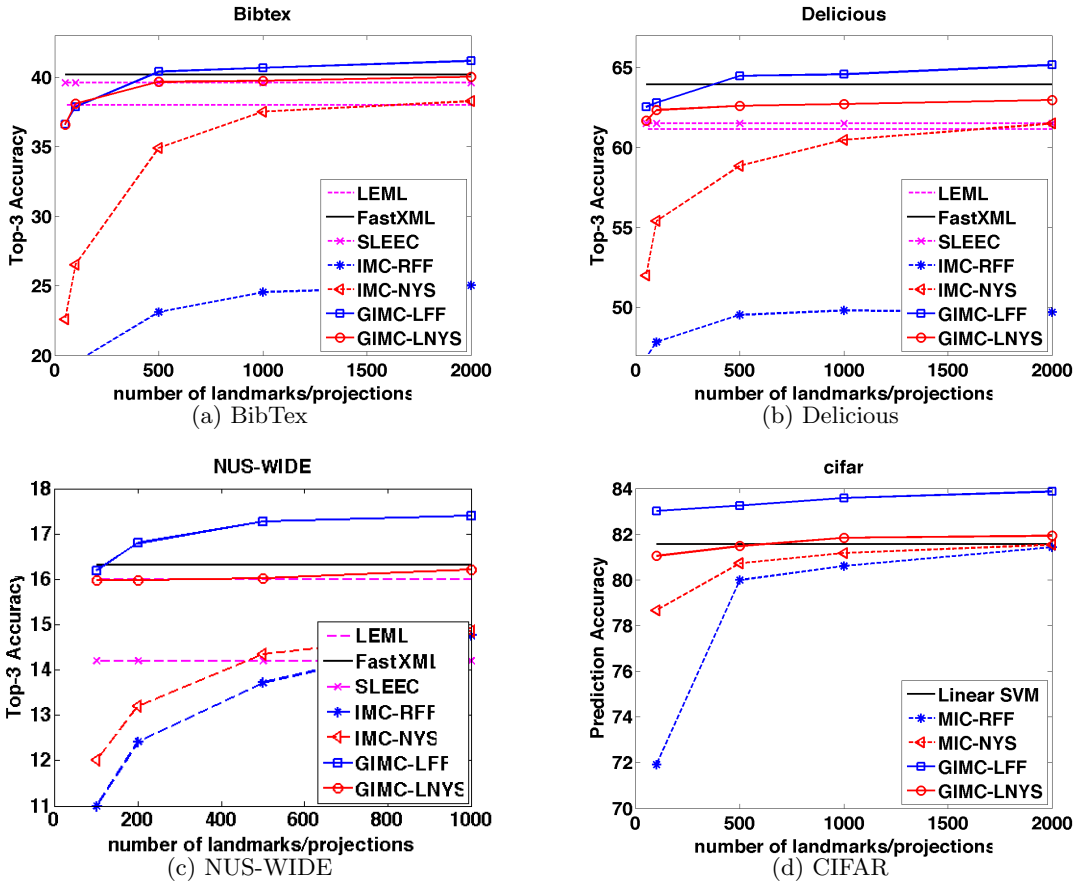
where π_i^* is the ground-truth cluster of item i.

The result is shown in Table 3. First, we observe that MCCC, IMC-RFF, and GIMC-LFF all perform well on Mushroom as perfect clustering could be (almost) derived. This confirms that MCCC is indeed effective when clustering is linearly dependent on features (100% training classification accuracy could be attained by a linear SVM on this dataset). However, in Segment and Covtype where features are not linearly separable for clustering, MCCC is trapped and both IMC-RFF and GIMC-LFF perform better than MCCC on these two datasets. This shows that considering nonlinear mapping of features improves the clustering result when data points are not linearly separable. Furthermore, we see that GIMC-LFF can further improve IMC-RFF with sufficient constraints. This is because when constraints are sufficient, GIMC-LFF could reliably *learn* a better nonlinear mapping based on constraints, and thus a better clustering could be obtained. The results show that our framework outperforms both state-of-the-art MCCC algorithm and IMC-RFF by learning a better nonlinear mapping of features for clustering.

7. CONCLUSION

We propose a family of Goal-directed Inductive Matrix Completion (GIMC) algorithms. Instead of constructing non-linear features without any supervision to the final goal, we formulate the non-linear mapping finding stage and the model learning stage together as a unified function. As a result, out algorithms find the non-linear mapping that is good for Inductive Matrix Completion. We apply our two algorithms, GIMC based on Fourier features (GIMC-LFF) and GIMC based on Nyström features(GIMC-LNYS), to multi-label learning, multi-class classification, and semi-supervised clustering problems. Experimental results show that our algorithms significantly reduce the number of non-linear features needed, and we obtain a better prediction accuracy and lower clustering error rate compared to state-of-the-art methods.

[5]We subsample from the entire dataset to make clusters have balanced size.



(a) BibTex (b) Delicious

(c) NUS-WIDE (d) CIFAR

Figure 2: The comparison on multi-label and multi-class classification problems. Figures 2(a),2(b), and 2(c) show results for multi-label learning problem and Figure 2(d) shows results for multi-class classification problem. x-axis shows the number of projections in GIMC-LFF or number of landmark points in GIMC-LNYS. y-axis shows the Top-3 accuracy for multi-label learning or prediction accuracy for multi-class classification. In Figure 2(b) and 2(c) we can observe GIMC-LFF and GIMC-LNYS are much better than IMC-RFF and IMC-NYS for multi-label learning. GIMC-LFF even outperforms state-the-art multi-label solvers (LEML, FastXML, and SLEEC). In Figure 2(d), our algorithms also have good performance when applying to multi-class problems such as the CIFAR-10 dataset.

As future work, we are planning to exploit (1) different non-linear features mapping, e.g., polynomial kernel feature mapping; (2) different optimization techniques in our framework to speed up the computation, e.g., using coordinate descent and stochastic gradient descent; (3) other machine learning applications, such as classification and regression.

Acknowledgments

This research was supported by NSF grants CCF-1320746 and IIS-1546459. We also thank Dinesh Jayaraman for help on CAFFE installation, Hsiang-Fu Yu for help on LEML code, Prateek Jain and Manik Varma for sharing the code and parameters for FastXML and SLEEC.

References

[1] K. Bhatia, H. Jain, P. Kar, M. Varma, and P. Jain. Sparse local embeddings for extreme multi-label classification. In *NIPS*. 2015.

[2] K.-Y. Chiang, C.-J. Hsieh, and I. S. Dhillon. Matrix completion with noisy side information. In *NIPS*, 2015.

[3] B. Dai, B. Xie, N. He, Y. Liang, A. Raj, M. Balcan, and L. Song. Scalable kernel methods via doubly stochastic gradients. In *NIPS*, 2014.

[4] C. Hsieh, K. Chiang, and I. S. Dhillon. Low rank modeling of signed networks. In *KDD*, pages 507–515, 2012.

[5] P. Jain and I. S. Dhillon. Provable inductive matrix completion. *CoRR*, abs/1306.0626, 2013.

[6] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[7] P. Kar and H. Karnick. Random feature maps for dot product kernels. *arXiv preprint arXiv:1201.6530*, 2012.

[8] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *IEEE Computer*, 42(8):30–37, 2009.

Dataset	n	d	k	# constraints $\|\Omega\|$	k-means	MCCC	IMC-RFF	GIMC-LFF
Mushroom	8142	112	2	n	0.1897	0.0007	0.0034	**0.0002**
				$5n$	0.1897	0.0010	0.0005	**0.0000**
				$10n$	0.1897	0.0007	**0.0000**	**0.0000**
				$15n$	0.1897	0.0007	0.0002	**0.0000**
				$20n$	0.1897	0.0007	0.0002	**0.0000**
Segment	2319	19	7	n	0.1433	0.0891	**0.0683**	0.0724
				$5n$	0.1347	0.0800	0.0580	**0.0570**
				$10n$	0.1363	0.0809	0.0650	**0.0446**
				$15n$	0.1362	0.0872	0.0678	**0.0402**
				$20n$	0.1330	0.0837	0.0564	**0.0380**
Covtype-sub	1711	54	7	n	0.2523	0.2498	**0.1840**	0.1898
				$5n$	0.2112	0.1772	0.1930	**0.1592**
				$10n$	0.2068	0.1708	0.1722	**0.1388**
				$15n$	0.2203	0.1677	0.1687	**0.1262**
				$20n$	0.2124	0.1607	0.1561	**0.1078**

Table 3: Semi-supervised clustering results. Here n is the number of data points, d is the dimension of the data, and k is the number of clusters. In this table, we vary $|\Omega|$ pairwise constraints with $|\Omega| = [1, 5, 10, 15, 20] \times n$ and report the clustering error rate achieved by each method. We can see that our proposed method GIMC-LFF achieves the best performance among almost all settings in these three datasets. This shows that learning non-linear mappings and models jointly will benefit IMC.

[9] A. Krizhevsky. Learning Multiple Layers of Features from Tiny Images. Master's thesis, 2009.

[10] R. Kueng, H. Rauhut, and U. Terstiege. Low rank matrix recovery from rank one measurements. *Applied and Computational Harmonic Analysis*, 2015.

[11] S. Kumar, M. Mohri, and A. Talwalkar. Ensemble Nyström methods. In *NIPS*, 2009.

[12] Q. V. Le, T. Sarlos, and A. J. Smola. Fastfood – approximating kernel expansions in loglinear time. In *ICML*, 2013.

[13] H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King. In *WSDM*, pages 287–296.

[14] N. Natarajan and I. S. Dhillon. Inductive matrix completion for predicting gene-disease associations. *Bioinformatics*, 30(12):60–68, 2014.

[15] J. Oh, W.-S. Han, H. Yu, and X. Jiang. Fast and robust parallel sgd matrix factorization. In *KDD*, 2015.

[16] Y. Prabhu and M. Varma. Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning. In *KDD*, 2014.

[17] A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *NIPS*, 2007.

[18] A. Rahimi and B. Recht. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. *NIPS*, 2009.

[19] N. Rao, H.-F. Yu, P. Ravikumar, and I. S. Dhillon. Collaborative filtering with graph information: Consistency and scalable methods. In *NIPS*, 2015.

[20] B. Recht and C. Ré. Parallel stochastic gradient algorithms for large-scale matrix completion. *Mathematical Programming Computation*, 5(2):201–226, 2013.

[21] B. Recht, C. Re, S. Wright, and F. Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *NIPS*, pages 693–701. 2011.

[22] N. Srebro and A. Shraibman. Rank, trace-norm and max-norm. In *COLT*, pages 545–560, 2005.

[23] C. K. I. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *NIPS*, 2001.

[24] M. Xu, R. Jin, and Z.-H. Zhou. Speedup matrix completion with side information: Application to multi-label learning. In *NIPS*, pages 2301–2309, 2013.

[25] J. Yi, L. Zhang, R. Jin, Q. Qian, and A. Jain. Semi-supervised clustering by input pattern assisted pairwise similarity matrix completion. In *ICML*, pages 1400–1408, 2013.

[26] H.-F. Yu, C.-J. Hsieh, S. Si, and I. S. Dhillon. Scalable coordinate descent approaches to parallel matrix factorization for recommender systems. In *ICDM*, 2012.

[27] H.-F. Yu, C.-J. Hsieh, S. Si, and I. S. Dhillon. Parallel matrix factorization for recommender systems. *Knowledge and Information Systems (KAIS)*, 41(3):793–819, 2014.

[28] H. F. Yu, P. Jain, P. Kar, and I. S. Dhillon. Large-scale multi-label learning with missing labels. In *ICML*, 2014.

[29] K. Zhang, I. W. Tsang, and J. T. Kwok. Improved Nyström low rank approximation and error analysis. In *ICML*, 2008.

[30] T. Zhou, H. Shan, A. Banerjee, and G. Sapiro. Kernelized probabilistic matrix factorization: Exploiting graphs and side information. In *SDM*, 2012.