One-Class Matrix Completion with Low-Density Factorizations

Vikas Sindhwani*, Serhat S Bucak†, Jianying Hu* and Aleksandra Mojsilovic*
*Business Analytics and Mathematical Sciences
IBM T.J. Watson Research Center, Yorktown Heights, NY 10598
Email: {vsindhw,jyhu,aleksand}@us.ibm.com
†Department of Computer Science and Engineering, Michigan State University, East Lansing, MI 48824
Email: bucakser@msu.edu

***Abstract*—Consider a typical recommendation problem. A company has historical records of products sold to a large customer base. These records may be compactly represented as a sparse customer-times-product "who-bought-what" binary matrix. Given this matrix, the goal is to build a model that provides recommendations for which products should be sold next to the existing customer base. Such problems may naturally be formulated as collaborative filtering tasks. However, this is a *one-class* setting, that is, the only known entries in the matrix are one-valued. If a customer has not bought a product yet, it does not imply that the customer has a low propensity to *potentially* be interested in that product. In the absence of entries explicitly labeled as negative examples, one may resort to considering unobserved customer-product pairs as either missing data or as surrogate negative instances. In this paper, we propose an approach to explicitly deal with this kind of ambiguity by instead treating the unobserved entries as optimization variables. These variables are optimized in conjunction with learning a weighted, low-rank non-negative matrix factorization (NMF) of the customer-product matrix, similar to how Transductive SVMs implement the low-density separation principle for semi-supervised learning. Experimental results show that our approach gives significantly better recommendations in comparison to various competing alternatives on one-class collaborative filtering tasks.**

***Keywords*-Collaborative Filtering; NMF; Implicit Feedback; Matrix Completion**

I. Introduction

Recommender systems have become increasingly important tools to help users efficiently sort through a large number of offered items and services and focus on the ones that are mostly likely of interest. Broadly speaking, recommender systems are typically based on one of two alternative strategies. The *content based approach* creates profiles that capture the characteristic features of users and items, and uses these features to identify likely linkage/affinity between a user and an item. The main difficulty with this approach is that it is often laborious and some times impossible to collect the external information needed to create the profiles. In contrast, the *collaborative filtering* approach relies only on past user activities as recorded in transaction history or satisfaction ratings. Besides the reduced burden on data collection , another major appeal of collaborative filtering is that it is domain agnostic. For example, the same algorithm can be expected to apply equally well to consumers in an e-commerce setting (e.g., Amazon) and corporate customers of a large IT company (e.g., IBM).

The recently concluded million-dollar Netflix competition has catapulted collaborative filtering and, in particular, matrix factorization techniques, to the forefront of recommender technologies [1]. However, these methods rely on the availability of explicit feedback, and it is well known that their performance is bounded by the number of observed matrix entries. In a Netflix-like setting, the user-movie matrix consists of three kinds of entries: positive ratings expressing viewing preferences, negative ratings expressing dislike, and unrated movies that may be simply considered as missing data to be estimated. On the other hand, many application settings inherently generate one-class (i.e., positive only) datasets, since the knowledge required for labeling examples as negative is typically not available explicitly and difficult to collect. In such cases, the observed events are reliable indicators of what the user liked. However, there is no explicit information about what the user did not like, because the unobserved user-item pairs can be interpreted in many different ways. For example, the reason why a user did not purchase a product could be that she was simply not aware of it.

With positive-only data, matrix factorization models may be learnt by treating zeros (unobserved entries) as missing data. This is an intuitively suboptimal strategy since it attempts to learn only from a very small set of positive examples. At the other extreme is the strategy of treating zeros as negative. This too seems suboptimal in that a user-item pair that may turn positive in the future is marked as a low-affinity (negative) example. The latter methodology does have the advantage that if most zeros are indeed negative, then the latent factors provide a representation where high-affinity user-item pairs can be better discriminated against the low-affinity ones, modulo labeling errors that are introduced by marking *all* zeros as negative.

We formulate a new strategy that avoids either extreme by means of explicit optimization. We treat the associated user-item pair as an optimization variable. The latent factors and these discrete label variables are learnt simultaneously. We propose a novel procedure to minimize the associated objective function, drawing from global optimization

techniques (deterministic annealing/continuation/homotopy methods) for combinatorial and non-convex problems. These techniques have been utilized in the context of Low-density separation in Semi-supervised SVMs [2], [3]; but here, they are applied in conjunction with NMF optimization.

II. Background and Related Work

A large number of techniques have been proposed for collaborative filtering (see, e.g., [4], [5], [6], [7], [8]), with some extensions to incorporate additional user-item attributes [9], [10]. In a typical matrix factorization approach to collaborative filtering, a customer and a product are represented as unknown feature vectors $w, h \in \mathbf{R}^k$ whose dimensions are considered as k latent factors. These feature vectors are learnt so that inner products $w^T h$ match the known preference ratings. This is equivalent to the problem of building *weighted* approximations of the preference matrix where weights are chosen such that known ratings are emphasized in measuring the quality of the approximation. Various models (see [11], [1]) differ in the approximation criteria or the loss function they employ, and the kinds of regularization used to avoid overfitting.

One-class matrix factorization is a relatively recent theme of research, despite the ubiquity of recommendation tasks where they could be used. [12], [13], [14] are recent papers that propose weighting and sampling schemes to handle one-class settings with unconstrained factorizations[1] based on the squared loss. The essential idea is to treat all non-positive user-item pairs as negative examples, but appropriately control their contribution in the objective function via either uniform, user-specific or item-specific weights. Our formulation in this paper subsumes these ideas as special cases. Preliminary experiments show that our proposed method outperforms the global, user and item weighting schemes suggested by [12].

Several recent papers [15], [16], [17] have considered the general problem of *Matrix Completion* from few observed entries. A surprising recent result [15], [16] states that an unknown low-rank matrix can be exactly recovered, under certain conditions, by solving the convex optimization problem of finding, among all matrices consistent with the observed entries, the one with minimum nuclear norm (sum of singular values). Note that these results do not apply to one-class settings where a rank-one matrix fits the observed matrix entries perfectly. This makes the one-class matrix completion problem radically different, and one that needs additional assumptions beyond low-rank to be approximately recovered. We introduce the notion of *low-density* assumption (i.e., the cluster assumption in the semi-supervised classification [18]), in addition to low-rank, to be able to address one-class settings. Under this constraint, one factor encodes low-density linear separators with respect to the point cloud induced by the other factor.

[1]Note: by "matrix factorization" we typically mean "matrix approximation" since exact factorization is not the goal here.

III. Formulation

Let X be a $m \times n$ binary matrix, such as a typical who-bought-what customer-product matrix. The set of non-zeros, $L = \{(i,j) : X_{ij} = 1\}$, denotes customer-product purchases. $X_{ij} = 0$ means that no purchase was made, but is not strictly a negative example. We will use $U = \{(i,j) : X_{ij} = 0\}$ to denote these "unlabeled" examples.

We assume that customers and products can be represented in an unknown lower-dimensional feature space, where features correspond to latent variables. Let $W = [w_1, \ldots, w_m]^T$ be an $m \times k$ matrix whose i^{th} row, w_i, is the k^{th} dimensional representation of a customer. Similarly, let $H = [h_1, \ldots, h_n]$ be a $k \times n$ matrix whose j^{th} column, h_j, is the k^{th} dimensional representation of a product. Then, weighted non-negative matrix factorization (WNMF) solves:

$$\underset{W\geq 0, H\geq 0}{\arg\min}\ \lambda\|W\|_F^2+\gamma\|H\|_F^2+\sum_{(i,j)\in L} C_{ij}V(X_{ij}, w_i^T h_j) \quad (1)$$

where V is a loss function (i.e., squared loss or generalized KL-divergence). For flexibility, we allow entry specific costs $C_{ij} \geq 0$. The real-valued parameters $\gamma \geq 0$ and $\lambda \geq 0$ tradeoff the regularizers against the data-fit terms. The above problem can also be solved without non-negativity constraints; however, learning non-negative factors is natural for non-negative data and lends "part-based" interpretability to the model [19], [20] . After learning W, H, the data matrix is reconstructed as $\hat{X} = WH$. The (i,j) customer-product pair for which $\hat{X}_{ij}$ is large are then recommended.

In a one-class setting, the loss function runs only over (i,j) pairs such that $X_{ij} = 1$. Since the loss function does not include zero-valued pairs, this corresponds to treating zeros as missing values. We refer to this approach as ZAM (zeros-as-missing) approach. An alternative approach is to treat zeros as negative examples, ZAN (zeros-as-negative). Note that the ZAN model is biased towards producing low-scores for products that a customer has not bought before, which may not be an accurate assumption. In this paper, we consider an alternative between ZAM and ZAN. We call it $ldNMF$, which stands for *low-density non-negative matrix factorizations*, given the conceptual similarity to low-density methods in semi-supervised earning [18]. The $ldNMF$ problem can be formulated as:

$$\begin{aligned}\underset{\substack{W\geq 0, H\geq 0\\ y_{ij}\in\{0,1\},(i,j)\in U}}{\arg\min}\quad & J\left(W, H, \{y_{ij}\}_{(i,j)\in U}\right) = \\ \lambda\|W\|_F^2 + \gamma\|H\|_F^2 + & \sum_{(i,j)\in L} C_{ij}V(X_{ij}, w_i^T h_j) \\ & + \sum_{(i,j)\in U} C_{ij}V(y_{ij}, w_i^T h_j) \qquad (2)\end{aligned}$$

where $J\left(W, H, \{y_{ij}\}_{(i,j)\in U}\right)$ is the objective function whose first two optimization variables are the latent factors, W, H, while the third set of variables are discrete $\{0, 1\}$-valued variables, i.e., $y_{ij} = 1$ implies positive class while $y_{ij} = 0$ implies negative class.

We will solve the optimization problem of Eq. (2), subject to the constraint that a certain user-specified fraction of the optimization variables are positive, i.e. $\frac{1}{|L|}\sum_{(i,j)\in U} y_{ij} = 2r - 1$, where r will be a user-specified parameter which we will refer to as the *positive class ratio*. Similar constraints are added in the formulations for Semi-supervised SVMs [18]. Note some special cases of $ldNMF$. If we set $r = 0$, then $y_{ij} = 0, (i, j) \in U$, we are lead back to the ZAN model. When we set $C_{ij} = 0, (i, j) \in U$, the $ldNMF$ model trivially reduces to ZAM.

IV. Optimization Algorithms

We propose a simple alternating minimization algorithm. For any fixed setting of the y_{ij} variables, the sub-problem of optimizing W, H is a weighted NMF and a large family of techniques can in principle be brought to bear here (see [20] for a review).

The other sub-problem, that of optimizing $y_{ij}, (i, j) \in U$ keeping W and H fixed, is a discrete optimization problem. There are two problems to address; *(i)* we need to additionally satisfy class balance constraint, and *(ii)* by aggressively committing to discrete labels early in the optimization, the procedure runs the risk of getting trapped in a sub-optimal local minima. To address the latter issue, we focus on deterministic annealing/homotopy methods given their robustness to presence of sub-optimal local minima. These are well-known techniques for handing discrete optimization variables. We point the reader to [3], [2] for an overview. Operationally speaking, they involve the following steps:

1) Relax discrete variables y_{ij} to real valued probability-like variables p_{ij}. Instead of optimizing $J(W, H, \{y_{ij}\}_{ij\in U})$ with respect to y_{ij}, optimize the expected value of it under the probabilities p_{ij}.
2) Smooth the new objective function, such that as the smoothing parameter is varied, we solve a sequence of optimization problems of increasing difficulty where the solution of an easier optimization problem is used as the starting point of a harder optimization. This kind of smoothing protects against local minima.

Let p_{ij} denote the probability that $y_{ij} = 1$. The modified optimization problem is the following,

$$\begin{aligned}
&\underset{W\geq 0, H\geq 0, \{p_{ij}\}_{(i,j)\in U}}{\arg\max} \quad J_T(W, H, \{p_{ij}\}_{ij\in U}) = \\
&\lambda\|W\|_F^2 + \gamma\|H\|_F^2 + \sum_{(i,j)\in L} C_{ij} V(1, w_i^T h_j) \\
&+ \sum_{(i,j)\in U} C_{ij}\left(p_{ij} V(1, w_i^T h_j) + (1 - p_{ij}) V(0, w_i^T h_j)\right) \\
&- T \sum_{(i,j)\in U} H(p_{ij}) \quad \text{subject to: } \frac{1}{|U|}\sum_{ij} p_{ij} = r \quad (3)
\end{aligned}$$

The third line in the equation above represents the expected loss under the probabilities p_{ij}. The last term $H(p) = -p\ log(p) - (1 - p)\ log(1 - p)$ is the smoothing function measuring entropy. When T is very high, entropy is maximized at $p_{ij} = r$. This corresponds to essentially solving a softer version of ZAN (letting the negative label be r instead of 0). As T is decreased, the optimal p_{ij} can be shown to progressively harden to discrete variables.

We outline an alternating optimization procedure to minimize $J_T(\cdot, \cdot, \cdot)$. First, let us assume T is fixed. Our block descent procedure first optimizes W and H (NMF), while keeping p_{ij}'s fixed. Then keeping W, H fixed, we optimize p_{ij}'s under the class ratio constraint. This is a convex problem that can be solved exactly.

A. Optimizing W, H for fixed p_{ij} variables

For fixed p, the optimization over W, H involves the first four terms of Eq. (3). Adding a constant term $\sum_{ij} C_{ij} p_{ij}^2$, the forth term of Eq. (3) can be expressed as $\sum_{ij} C_{ij}(p_{ij} - w_i^T h_j)^2$. Thus, it is easy to see that W and H can be obtained by solving,

$$\underset{W,H}{\arg\min}\ \lambda\|W\|_F^2 + \gamma\|H\|_F^2 + \|C^{\otimes 0.5} \otimes \left(\hat{X} - WH\right)\|_F^2 \quad (4)$$

where we use the following notation: $\hat{X} = (X + P)$, $A \otimes B$ denotes elementwise product between matrices A and B, $(C^{\otimes 0.5})_{ij} = \sqrt{C_{ij}}$, P is the matrix of optimization variables, whose elements equal p_{ij} when $(i, j) \in U$ and 0 when $(i, j) \in L$. Thus, the sub-problem of minimizing W, H for fixed p_{ij}'s is the weighted NMF problem of Eq. (4). The solution can be obtained by alternating between the following two multiplicative update steps,

$$H = H \otimes \frac{W^T\left[C \otimes (X + P)\right]}{W^T\left[C \otimes (WH)\right] + \gamma H} \quad (5)$$

$$W = W \otimes \frac{\left[C \otimes (X + P)\right] H^T}{\left[C \otimes (WH)\right] H^T + \lambda W} \quad (6)$$

where division is elementwise. Once inside this sub-routine, steps in Eqs. (5) and (6) are repeatedly performed until relative improvement in the NMF objective function falls below some user-specified tolerance, or a maximum number of iterations are exceeded.

B. Optimizing p_{ij} variables for fixed W, H

For fixed W, H, the optimization over p_{ij} involves the fourth and fifth term in the objective function of Eq. (3), subject to the balance constraint. Let ν be the Lagrange multiplier corresponding to the balance constraint, $\frac{1}{|U|}\sum_{ij} p_{ij} = r$. By defining $g_{ij} = S_{ij}\left[V(1, o_{ij}) - V(0, o_{ij})\right]$, forming the

Lagrangian and settings its gradient to 0, the optimal p_{ij} can be shown to be given by,

$$p_{ij} = \frac{1}{1 + e^{\frac{g_{ij} - \nu}{T}}} \tag{7}$$

where ν can be found by substituting the above in the balance constraint and solving:

$$\frac{1}{|U|} \sum_{ij \in U} \frac{1}{1 + e^{\frac{g_{ij} - \nu}{T}}} = r \tag{8}$$

The root is computed by using a hybrid combination of Newton-Raphson iterations and the bisection method together with a carefully set initial value [3].

C. Complexity & Large-Scale Implementation

In this subsection, we first examine the complexity of a naive implementation. Then we show how special structure of the cost matrix C together with sparsity in X, and choice of a sparse set of optimization variables P allow us to scale to very large datasets. We use the notation $nz[A]$ to denote the number of non-zeros in the matrix A.

Consider the update of H in Eq. (5). The overall complexity is $O(nz[X + P]k + mnk)$ and is clearly prohibitive for large m, n requiring also the computation of a large dense intermediate matrix WH. This complexity calculation assumes that (a) there is no structure in C that can exploited for more efficient computations, (b) the matrix $X + P$ is dense. Now we show how to scale up by relaxing these assumptions:

• We assume P is p-sparse i.e., has p non-zero entries. This corresponds to only optimizing a subset of p_{ij} variables and fixing the rest to zero values. In this paper, for large-scale experiments, we take a random subset to be optimization variables though a judicious choice for it is an interesting technical problem.

• We assume C is an arbitrary low-rank non-negative matrix of the form $C = \sum_{i=1}^{q} \delta_i \phi_i^T$, where $\forall i : \quad \delta_i$ is a column vector of length m, $\forall i : \quad \phi_i$ is a column vector of length n, and $q \ll min(m, n)$ is the rank of C. The way we can utilize this structure is due to an easy-to-see connection between Hadamard (elementwise) products and Rank-one matrices: $C \otimes F = \sum_{i=1}^{q} D_{\delta_i} F D_{\phi_i}$ where F is any $m \times n$ matrix and the notation D_v implies is a diagonal matrix with diagonal elements equal to the elements of the vector v. In practice, we want to apply these weights only to unlabeled entries and retain a weight of 1 for labeled positive entries. This can be achieved as follows,

$$C \otimes F = \sum_{i=1}^{q} D_{\delta_i} F D_{\phi_i} - D_{\delta_i}(C_L \otimes F) D_{\phi_j} + (C_L \otimes F)$$

where C_L is 1 for positive entries and 0 otherwise. In other words, $C_L = X$. When $F = (X + P)$, this computation can be carried out in $O(qp)$ steps for the numerator. In the denominator, we make use of the following rearrangement:

$$W^T (C \otimes (WH)) = \sum_{i=1}^{q} \left(W^T D_{\delta_i} W\right) H D_{\phi_i}$$
$$- W^T D_{\delta_i} (C_L \otimes (WH)) D_{\phi_j} + W^T (C_L \otimes (WH)) \tag{9}$$

$$(C \otimes (WH)) H^T = \sum_{i=1}^{q} D_{\delta_i} W (H D_{\phi_i} H^T)$$
$$- D_{\delta_i} (C_L \otimes (WH)) D_{\phi_j} H^T + (C_L \otimes (WH)) H^T \tag{10}$$

The key observation is that since C_L is sparse, $C_L \otimes (WH)$ is a sparse matrix whose computation only requires the product WH to be evaluated where C_L is non-zero. Thus, this is a $O(pk)$ operation. The overall complexity is $O((m+n)qk^2 + qpk)$ which is much smaller than the naive implementation.

Solving the one-dimensional root finding problem in Eqn. 8 has negligible cost relative to the weighted NMF updates.

In summary, the use of low-rank cost matrices, C in conjunction with a sparse set of optimization variables leads to large-scale implementations. In practice, we use the rank-one setting where δ_1 gives weights for users and ϕ_1 gives weights for items as in [13], [14].

In Section V, we report the performance of $ldNMF$ at a fixed value of T, and study the sensitivity to this choice with respect to recommendation quality. In a full annealing implementation (not reported in this paper), T is gradually reduced to 0. Our convergence criteria is based on relative difference between KL-divergence of $p_i j$ variables between successive iterations, as also used in [2].

V. Empirical Study

Comparisons with Other One-Class Approaches: The first baseline method we are using, *Popularity*, is based on popularity of the items among users and the number of each user's past purchases. We mentioned the next two (*ZAM* and *ZAN*) earlier in the paper as natural one-class approaches. We also use three types of weighted *ZAN*: (i) wZAN (unif): a weighted version of ZAN where zeros are treated as negatives, but a uniform weight with value less than 1 is additionally imposed, (ii) wZAN (item-oriented): column weighting on the user-item matrix, (iii) wZAN (user-oriented): row weighting on the user-item matrix. These weighted ZAN schemes were proposed in [12], [13] and implemented with unconstrained factorizations; here, we apply them with NMF.

Evaluation Protocol: Recommender Systems typically show great variability with respect to choice of evaluation measure [21]. One-Class Collaborative Filtering experiments in [13] showed that no single technique tends to dominate with respect to all metrics. In this paper, we considered two

Table I
COMPARISON OF ALL METHODS IN TERMS OF AREA UNDER ROC CURVE AND AREA UNDER PRECISION-RECALL CURVE. NOTE THAT THE PARAMETERS OF BASELINE METHODS ARE CAREFULLY OPTIMIZED. $ldNMF$ GIVES STATISTICALLY SIGNIFICANT IMPROVEMENTS IN ALL CASES.

Methods		AUC-ROC			AUC-PrRc	
	$rank = 5$	$rank = 10$	$rank = 15$	$rank = 5$	$rank = 10$	$rank = 15$
Popularity	49.5 ± 0.4	49.5 ± 0.4	49.5 ± 0.4	0.9 ± 0.01	0.9 ± 0.01	0.9 ± 0.01
ZAM	41.6 ± 3.7	42.3 ± 4.2	41.6 ± 3.7	0.9 ± 0.01	0.9 ± 0.01	0.9 ± 0.01
ZAN	72.4 ± 0.4	73.5 ± 0.4	73.1 ± 0.3	0.9 ± 0.01	0.9 ± 0.01	0.9 ± 0.01
wZAN (unif)	72.6 ± 0.5	73.6 ± 0.4	73.3 ±0.3	20.9 ±0.2	22.7 ± 0.3	22.3 ± 0.2
wZAN (item)	72.9 ± 0.5	73.9 ± 0.3	73.7 ± 0.34	20.9 ±0.2	22.8 ± 0.3	22.5 ± 0.3
wZAN (user)	71.5 ± 0.26	72.0 ± 0.3	71.6 ± 0.3	11.5 ±0.1	13.1 ± 0.2	13.0 ± 0.2
ldNMF (proposed)	**74.8 ± 0.3**	**75.2 ± 0.3**	**74.9 ± 0.2**	**21.4 ± 0.3**	**23.3 ± 0.3**	**23.3 ± 0.4**

Table II
COMPARISON OF ALL METHODS IN TERMS OF TRAINING TIME (SEC)

Methods ↓ Rank →	5	10	15
Popularity	17	17	17
ZAM	330	400	375
ZAN	687	1044	1589
wZAN (unif)	747	868	1591
wZAN (item)	1056	1301	1982
wZAN (user)	703	881	1600
ldNMF (proposed)	1531	1785	1788

Table III
SENSITIVITY TO r AND T

r	0.001	0.01	0.1	0.3	0.5	0.7
AUC-PrRc	23.1	23.3	23.5	22.8	21.0	16.2
AUC-ROC	73.9	74.7	74.5	71.0	71.0	66.2
T	0.1	1	10	50	100	1000
AUC-PrRc	7.6	23.5	23.5	23.5	23.5	23.5
AUC-ROC	72.4	74.6	74.5	74.5	74.5	74.5

evaluation metrics: area under precision-recall curve (AUC-PrRc) and area under ROC curves (AUC-ROC). We also carefully optimized the baselines approaches over a large set of parameters and then observed whether the proposed method could further lead to performance improvements.

Small Scale Experiments on MovieLens: We first conducted experiments on the MovieLens dataset publically available at: `http://www.grouplens.org/`. The data consists of 100,000 ratings on an integer scale from 1 to 5 given to 1642 movies by 943 users. For one-class experiments, we removed all 3 and below ratings, and relabeled ratings 4 and 5 as 1, to then pose the task of recommending movies given user preferences alone. We created random training-test splits of positive customer-movie pairs in the ratio 75%-to-25% respectively. All results reported in this section are averaged over 10 random splits. We also use this dataset for a detailed study of comparison against baseline methods and sensitivity to parameters.

In Tables I and V we report AUC-PrRc, AUC-ROC and the training times respectively, for each of the 6 baselines, and compare them with $ldNMF$ for three choices of rank. For simplicity, for all methods, we chose $\gamma = \lambda = 0$. All baseline methods were initialized from the same initial random W, H, and the number of matrix factorization iterations are fixed.

As expected, since it only uses a small set of positive examples, ZAM returns the worst performance. Popularity also does not give good results. ZAN performs substantially better than ZAM and Popularity-based schemes. The performance of ZAN tends to improve with user-oriented weighting, but becomes comparable with item-oriented and uniform weightings. *$ldNMF$ gives statistically significant improvements for both precision-recall and ROC evaluation with only a small increase in overall training time.*

We report performance sensitivity to r in Table III (first 3 rows) keeping T fixed at 10. Similarly, keeping r fixed at 0.1, in Table III (last 3 rows) we report performance sensitity with respect to choice of T. We see that $ldNMF$ tends to be robust to the selection of r, as the performance is stable for $r \in [0.001, 0.2]$. Large values of r clearly enforce incorrect priors and naturally lead to loss of accuracy. With regards to sensitivity with respect to choice of T, we see that $ldNMF$ tends to be very stable.

Large-Scale Experiments on Netflix Dataset: We conducted large-scale experiments on the Netflix Prize dataset where we considered the one class "who-rated-what" problem i.e., whether a user rated a movie (a KDD Cup 2007 task). The Netflix matrix represents $480,189$ users and 17770 movies with around 100 million ratings. We considered a sparsity setting where a random set of 20 million ratings are available in a training matrix while the remaining 80 million are used for evaluation. We implemented the modified update equations developed in Section IV-C where we took a massive set of 20 million ratings as optimization variables. Table V lists the performance obtained by various methods. Recall that Weighted ZAN treats zeros as negatives and imposes uniform, user or item specific weights on them. We apply the same weights in our large-scale algorithm to demonstrate the ability to incorporate low-rank cost matrices.

Table V shows the computational performance on the netflix dataset in terms of observed increasing improvements in AUC-ROC as the optimizations progresses over time for $ldNMF$ (with a uniform cost matrix). The experiments were conducted on a cluster with nodes having ordinary CPU/memory configurations. These results clearly show that

Table IV
PERFORMANCE ON NETFLIX DATASET

Method	AUC-ROC
ZAM	51.3
ZAN	74.4
wZAN (uniform)	77.3
wZAN (item)	75.5
wZAN (user)	93.8
ldNMF (uniform)	**96.1**
ldNMF (item)	93.8
ldNMF (user)	94.2

Table V
COMPUTATIONAL PERFORMANCE (SECS)

Time	AUC-ROC
277	94.4
1096	94.7
5535	95.0
8671	95.8
11778	96.1

$ldNMF$ can, in practice, be run on large datasets by utilizing efficient sparse matrix computations to optimize several millions of variables. Table V shows that a comparison to baseline methods for various choices of cost matrices. These results also demonstrate that optimizing a random subset of variables as outlined in section IV-C can be sufficient for obtaining improvements. We see consistent improvements in AUC-ROC.

VI. CONCLUDING COMMENTS

In this paper, we have presented a principled, novel optimization approach to one-class matrix completion and collaborative filtering problems. Knowing that the underlying matrix is low-rank is insufficient for approximate recovery in this setting, making it necessary to make additional assumptions. We have drawn, both in terms of intuitions and also in terms of algorithmic frameworks, from semi-supervised learning methodologies based on the low-density assumption. Our method jointly learns a non-negative matrix factorization model for collaborative filtering while optimizing for unknown discrete label variables using non-convex optimization techniques. Our approach gives statistically significant improvements over 6 competing alternatives for one-class collaborative filtering with non-negative matrix factorizations. We are currently studying the empirical behavior of our approach with respect to annealing, rank and regularization parameters. We also plan to extend comparisons to other real-world one-class collaborative filtering problems. Between various choices of the loss function, regularizers, alternative optimization strategies and case-studies in various applications, we believe that this topic allows for a rich research agenda.

We close with a word on real-world business deployment scenario based on our experience. Interpretability is a very important concern as sellers and marketers not only need to act on a recommendation, but also have some sense of how to "pitch" the product. An open research direction is how to extract interpretability from latent factors and provide meaningful explanations for why a customer ought to be sold a recommended product. Also, business evaluation of a recommender system is somewhat different from measures like precision and recall. Practical considerations such as expected revenue and time taken to close a transaction, as well as the "non-obviousness" are also important business factors in judging the value of a recommendation.

REFERENCES

[1] R. B. Yehuda Koren and C. Volinsky, "Matrix factorization techniques for recommender systems," in *IEEE Computer*, vol. 42 (8), pp. 30–37.

[2] V. Sindhwani, S. Keerthi, and Olivier, "Deterministic annealing for semi-supervised kernel machines," in *International Conference on Machine Learning*, 2006.

[3] V. Sindhwani and S. Keerthi, "Large scale semi-supervised linear svms," in *SIGIR*, 2006.

[4] D. Goldberg, D. Nichols, B. Oki, and D. Terry, "Using collaborative filtering to weave an information tapestry," in *Comm. ACM, 35*, 1992.

[5] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in *UAI*, 1998.

[6] D. Heckerman, D. M. Chickering, C. Meek, R. Rounthwaite, and C. Kadie., "Dependency networks for inference, collaborative filtering, and data visualization." in *Journal of Machine Learning Research*, vol. 1, 2000, pp. 49–75.

[7] J. Rennie and N. Srebro, "Fast maximum margin matrix factorization for collaborative prediction." in *ICML*, 2005.

[8] N. Srebro and T. Jaakkola, "Weighted low-rank approximations," in *ICML*, 2003.

[9] P. Melville, R. Mooney, and R. Nagarajan, "Content-boosted collaborative filtering for improved recommendations," in *AAAI*, 2002.

[10] T. E. J.-P. V. J. Abernethy, F. Bach, "A new approach to collaborative filtering: Operator estimation with spectral regularization," in *JMLR*, vol. 10, 2009, pp. 803–826.

[11] T. K.-I. N. A. S. Miklo Kurucz, Andras A. Benczur and B. Torma., "Who rated what: a combination of svd, correlation and frequent sequence mining." in *KDD*, 2007.

[12] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. Lukose, M. Scholz, and Q. Yang, "One-class collaborative filtering," in *ICDM*, 2008.

[13] R. Pan and M. Scholz, "Mind the gaps: Weighting the unknown in large-scale one-class collaborative filtering." in *KDD*, 2009.

[14] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *ICDM*, 2008.

[15] E. Candes and T. Tao, "The power of convex relaxation: Near optimal matrix completion," *IEEE Transactions on Information Theory*, vol. 56 (5), pp. 2053–2080, 2009.

[16] B. Recht, "A simpler approach to matrix completion," *Journal of Machine Learning Research (to appear)*, 2010.

[17] R. Mazumder, T. Hastie, and R. Tibshirani, "Spectral regularization algorithms for learning large incomplete matrices," *Journal of Machine Learning Research (JMLR)*, vol. 11(Aug), pp. 2287–2322, 2010.

[18] O. Chapelle, V. Sindhwani, and S. Keerthi, "Optimization techniques for semi-supervised support vector machines," *JMLR*, vol. 9, pp. 203–233, 2006.

[19] D. Lee and H. Seung, "Learning the parts of objects by non-negative matrix factorization," in *Nature*, 1999.

[20] A. Cichocki, R. Zdunek, A. Phan, and S. Amari, *Nonnegative Matrix and Tensor Factorizations*. Wiley, 2009.

[21] A. Gunawardana and G. Shani, "A survey of accuracy evaluation metrics of recommendation tasks," in *Journal of Machine Learning Research*, vol. 10, 2009, pp. 2935–2962.