A Family of Non-negative Matrix Factorizations for One-Class Collaborative Filtering Problems

V. Sindhwani
Mathematical Sciences
IBM Research
vsindhw@us.ibm.com

S.S. Bucak
Dept. Of Computer Science
Michigan State University
ssbucak@us.ibm.com

J. Hu A. Mojsilovic
Mathematical Sciences
IBM Research
{jyhu,aleksand}@us.ibm.com

ABSTRACT

This paper is motivated by the industrial research problem of designing a real-world recommender system for a large Information Technology (IT) company. Given historical records of client purchases, compactly represented as a sparse client-times-product "who-bought-what" binary matrix, the goal is to build a model that provides recommendations for what products should be sold next to the existing client base. Such a problem may naturally be formulated as a collaborative filtering task. However, this is a *one-class* setting, that is, if a client has not bought a product yet, it does not imply that the client has a low propensity to potentially buy that product later. In the absence of explicitly labeled negative examples, one may resort to considering zero-valued client-product pairs as either missing data or as surrogate negative instances. In this paper, we outline an approach to explicitly deal with this kind of ambiguity by instead treating zero-valued pairs as optimization variables. These variables are optimized in conjunction with learning a weighted, low-rank non-negative matrix factorization (NMF) of the client-product matrix. The proposed algorithm alternates NMF optimization with deterministic-annealing/continuation techniques designed for global minimization of combinatorial and non-convex objective functions. Experimental results show that our approach can give significantly better recommendations in comparison to various competing alternatives on a one-class collaborative filtering task.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Algorithms

Keywords

Collaborative Filtering, Non-negative Matrix Factorizations

RecSys '2009 New York, USA
Recommender Based Industrial Applications Workshop

1. INTRODUCTION

Large multinational IT organizations sell a huge variety of products to a large number of clients. The set of products is highly diverse ranging from hardware and software to business intelligence and outsourcing services. The client base is typically global, covering a wide range of firmographic characteristics. In such a large-scale business environment, there arises a need to make data-driven decisions on where to optimally allocate sales resources. A model that can analyze historical purchase patterns of existing clients, and make useful recommendations to sellers and marketers as to which clients should be targeted next with what products, can add immense value – not only from the perspective of day-to-day selling, but also in terms of supporting longer term business strategy formation by providing an accurate and dynamic view of opportunities that are likely to enter the sales pipeline once the recommendations are followed.

Purchase data is typically meticulously recorded in corporate databases. This data can be represented as an $m \times n$ binary matrix where *one*-valued entries represent purchases, and *zero*-valued entries represent products not yet purchased, by m clients over a space of n products. Detailed business knowledge of clients and products is typically spread in a diffused manner over the entire organization. It is therefore attractive to attempt to make sales recommendations without significant manual involvement, or without painstaking management of rapidly changing client and product profiles. *Collaborative filtering* is a natural choice for designing such a system since it generates recommendations by centrally analyzing the client-product matrix *alone*.

The recently concluded million-dollar Netflix competition has catapulted collaborative filtering and, in-particular, matrix factorization techniques to the forefront of recommender technologies [25, 1, 14]. Describing their prize-winning ensemble of techniques in a recent paper [25], the authors state that factorization models are "*the most popular and successful methods for predicting ratings*" constituting the majority of the models in their ensemble. Yet, when thinking of applying matrix factorization techniques to the sales recommendation problem, we realize that it is quite different from movie recommendation. 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, the sales recommendation setting intrinsically generates one-class datasets since the business domain knowledge for negative examples – that a client has no interest in buy-

ing a product ever in the future – is typically not available explicitly in corporate databases. Moreover, such knowledge is difficult to gather and maintain in the first place, given the rapidly changing business environment, and therefore, also, client tendencies to purchase a product.

A low-rank matrix factorization approach starts with the assumption that there exist a small number, say k, of latent factors that can explain the buying behavior of clients. Clients and Products are represented as feature vectors in this latent space, where similarity between a client-product pair models the propensity of the client to buy that product. While exact interpretability is never easy with latent factor models, one can imagine that latent factors implictly capture attributes such as firmographics for clients (e.g., size, turn-over, industry-sector) or "bundles" for product (e.g., packaged solutions combining hardware, software and services).

With positive-only data, matrix factorization models may be learnt by treating all zeros (no-purchases) as missing data. This is an intuitively suboptimal strategy since it attempts to learn from only a very small set of positive examples. At the other extreme is the strategy of treating all zeros as negative. This too seems suboptimal in that a client-product deal that is likely to materialize in the future, but has not done so yet, 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 client-product 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 one-class strategy that avoids either extreme by means of explicit optimization. Since we truly do not know the status of the interest of a client in a product they have yet not purchased, we treat the associated client-product pair as an optimization variable. The latent factors and these discrete label variables are learnt simultaneously, thus bringing the advantage of learning discriminative factors, while simultaneously attempting to judiciously assign labels. We propose a novel procedure to minimize the associated objective function, drawing from global optimization techniques (known as deterministic annealing/continuation/homotopy methods) for combinatorial and non-convex problems. These techniques have been previously utilized in the context of Semi-supervised SVMs [23, 22]; but here, they are applied in conjunction with NMF optimization. Experimental results on a one-class movie recommendation task show that the proposed technique can generate significantly better recommendations, in comparison to various competing alternatives.

2. BACKGROUND & RELATED WORK

In its purest form, the goal of Collaborative Filtering is to construct a model to predict preferences of clients ("users") over a range of products ("items"), based *only* on observations of historical purchases. A large number of techniques have been proposed (see, e.g., [7, 3, 8, 21, 24], and references therein) with some extensions to incorporate additional user-item attributes [17, 12]. In a typical matrix factorization approach to Collaborative Filtering, a client 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 by taking the product of low-rank factor matrices, where weights are chosen such that known ratings are emphasized in measuring the quality of the approximation. Once the factors are learnt, they provide a prediction for unknown ratings which may then be used for generating recommendations. Various models differ in the approximation criteria or the loss function they employ, and variants may be derived by adding different kinds of regularization to avoid overfitting. For such models in the context of the Netflix problem, see [1, 14, 18, 25].

Much of the work in collaborative filtering assumes availability of a range of high and low ratings, or multiple classes in the data matrix. One-class collaborative filtering is a relatively recent theme of research, despite the ubiquity of one-class recommendation tasks. [20, 19] 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 goal in this paper is to attempt a principled optimization over the labels of non-positive user-item pairs. Preliminary experiments show that our proposed method outperforms the global, user and item weighting schemes suggested by [20].

Much of the prior work in these contexts has explored unconstrained SVD-like factorizations, while we focus on the use of Non-negative Matrix Factorizations (NMF) [15, 5]. NMF imposes non-negativity constraints on the latent factors. The purchase history of each client is approximated as a mixture of product "parts" , i.e., basis vectors in the product space. By disallowing subtractive basis, non-negativity constraints lend a "part-based" interpretation to the predictive model. NMF with generalized KL-divergence loss is equivalent [6] to Probabilistic Latent Semantic Analysis [9] which has previously been used for Collaborative Filtering tasks [10].

Our methods are based on alternating optimization. In a subroutine, we solve a weighted NMF optimization problem for which we use a version of the multiplicative update rules originally proposed by Lee and Seung [15], though other NMF optimization techniques can also be used. The multiplicative update equations for the weighted, unregularized case have been worked out in [2]. In our implementation we also added regularization terms. Our methods can be easily modified to further impose sparsity in the latent factors [11] by using l_1-regularization. A separate sub-routine handles optimization variables associated with non-positive user-item pairs. Here, we draw on deterministic annealing techniques previously used in the context of Transductive SVMs [23, 22]. In many applications these techniques show greater robustness to presence of sub-optimal local minima. Several other non-convex optimization techniques can also be brought to bear on this problem and we point the reader to [4] and references therein. With various choices of loss functions, regularizers and optimization strategies, we obtain a family of non-negative matrix factorizations for one-class collaborative filtering problems.

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

3. FORMULATION

Let X be a $m \times n$ binary matrix, such as a typical who-bought-what client-product matrix. X is usually large and sparse. The set of non-zeros, $S = \{(i,j) : X_{ij} = 1\}$, denotes client-product purchases. $X_{ij} = 0$ means that no purchase was made, but is not strictly a negative example. We will use the notation $\bar{S} = \{(i,j) : X_{ij} = 0\}$ to denote the complement of S, i.e., client-product pairs representing no-purchases.

In the standard matrix factorization setting, we assume that clients 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 client. 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 (NMF) solves the following optimization problem,

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

where V is a loss function, such as squared loss $V(y,t) = \frac{1}{2}(t-y)^2$, or generalized KL-divergence $V(y,t) = y \log \frac{y}{t} - y + t$, or hinge loss and its variants for max-margin matrix factorizations [21]. 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, which leads to weighted SVD-like solutions for the squared loss. However, in many contexts, learning non-negative factors is very natural and lends "part-based" interpretability to the model [15, 5]. Note also that while we used Frobenius norm i.e., l_2 regularizers in our experiments, our framework and algorithms easily allow for sparsity-encouraging l_1 regularization i.e., penalization of $\|W\|_1 = \sum_{i,j} W_{ij}$ and $\|H\|_1 = \sum_{ij} H_{ij}$, with minor modifications to iterative update rules detailed later in this paper.

After learning W, H, the data matrix is reconstructed as $\hat{X} = WH$. The (i,j) client-product pair for which $\hat{X}_{ij}$ is large (taking out pairs where $X_{ij} = 1$) 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). In that case, we solve,

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

Above, recall the notation that $\bar{S}$ represents the complement of S, i.e., the (i,j) pairs corresponding to zero values in X. Note that the ZAN model is biased towards producing low-scores for products that a client has not bought before, which may not be an accurate assumption.

In this paper, we consider an alternative between ZAM and ZAN, which we call $ZaOV$ which stands for *zeros as optimization variables*. The $ZaOV$ optimization problem has the following form,

$$\underset{\substack{W \geq 0, H \geq 0 \\ y_{ij} \in \{0,1\}, (i,j) \in \bar{S}}}{\arg\min}\ J\left(W, H, \{y_{ij}\}_{(i,j) \in \bar{S}}\right) = \lambda \|W\|_F^2 + \gamma \|H\|_F^2 + \sum_{(i,j) \in S} C_{ij} V(1, w_i^T h_j) + \sum_{(i,j) \in \bar{S}} C_{ij} V(y_{ij}, w_i^T h_j) \quad (3)$$

where $J\left(W, H, \{y_{ij}\}_{(i,j) \in \bar{S}}\right)$ denotes the objective function above, 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. Let us compare Equation 2 with Equation 3. Note that the only difference is the last term. In the former case, we commit to all zero-valued entries in X to be treated as "negative examples" which biases the ZAN model towards producing low-scores for associated client-product pairs, while in the latter $ZaOV$ model, we declare that these pairs are truly uncertain and therefore the associated variables, y_{ij}, actually need to be optimized.

We will solve the optimization problem of Equation 3, subject to the constraint that a certain user-specified fraction of the optimization variables are positive:

$$\frac{1}{|\bar{S}|} \sum_{(i,j) \in \bar{S}} y_{ij} = r \quad (4)$$

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 Transductive SVMs [13, 4].

Note some special cases of $ZaOV$. If we set $r = 0$, then $y_{ij} = 0, (i,j) \in \bar{S}$, and the associated optimal values of W, H lead to the ZAN model. When we set $C_{ij} = 0, (i,j) \in \bar{S}$, the $ZaOV$ model trivially reduces to ZAM.

We now outline an alternating optimization algorithm to minimize the $ZaOV$ objective function in Equation 3 under the class balance constraint of Equation 4.

4. OPTIMIZATION ALGORITHMS

We propose a simple alternating minimization algorithm. First note that the $ZaOV$ objective function has a mix of continuous and discrete variables, i.e., it has two matrix-valued continuous variables, $W \in \mathbf{R}^{m \times k}$ and $H \in \mathbf{R}^{k \times n}$, and a collection of binary variables $y_{ij} \in \{0,1\}$. For any fixed setting of the y_{ij} variables, the sub-problem of optimizing W, H is a weighted non-negative matrix factorization. Simple modifications of standard NMF techniques can be used to solve this sub-problem. In-fact, alternating optimization is the typical workhorse here too since the optimization over W keeping H fixed, and vice versa, are convex problems. A large family of techniques can in principle be brought to bear here including Lee and Seung's multiplicative updates [15], projected gradient techniques [16] and a variety of other methods (see [5] for a review).

The other sub-problem, that of optimizing $y_{ij}, (i,j) \in \bar{S}$ keeping W and H fixed, is a discrete optimization problem.

It is easy to see that the optimal values are given by,

$$y_{ij} = \begin{cases} 0 & if\ V(0, w_i^T h_j) < V(1, w_i^T h_j) \\ 1 & otherwise \end{cases} \quad (5)$$

It is possible to alternate W, H optimization with the above settings for y_{ij}'s. However, there are two problems we need to address; one: we need to additionally satisfy class balance constraint of Equation 4, and two: 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 describe a more conservative approach using deterministic annealing. In order to satisfy the class balance constraints, Equation 4, one may implement a label switching procedure akin to Joachim's Transductive SVMs [13]. Starting from the ZAM solution, client-product pairs $(i, j) \in \bar{S}$ are ranked by $w_i^T h_j$ and the associated y_{ij} of the top $r|\bar{S}|$ are re-set to a value of 1. This gives a feasible initial setting for $y_{ij}, (i, j) \in \bar{S}$. Next, a sequence of ZAN problems are solved in alternation with a label switching sub-routine responsible for locally optimizing y_{ij} variables. Let (a, b) and (c, d) be two client-product pairs such that,

$$\begin{aligned} y_{ab} = 1, y_{cd} = 0 \\ C_{ab} V\left(1, w_a^T h_b\right) + C_{cd} V\left(0, w_c^T h_d\right) > \\ C_{ab} V\left(0, w_a^T h_b\right) + C_{cd} V\left(1, w_c^T h_d\right) \end{aligned} \quad (6)$$

If the above condition is satisfied, then swapping the labels, i.e., setting $y_{ab} = 0, y_{cd} = 1$ leads to a decrease in the $ZaOV$ objective function, while continuing to satisfy the balance constraint. After the label swap, ZAN is retrained with new labels, starting the optimization from the previous values of W and H leading to further decrease. For a faster multi-switch heuristic, and termination criteria, that can be used here, we point the reader to [22] where this procedure is implemented together with SVM optimization.

The label switching procedure briefly outlined above is one member of a family of optimization algorithms (see [4]) that can be used to solve Equation 3. In this paper, we focus on deterministic annealing/homotopy methods given their robustness to presence of sub-optimal local minima in many applications. These are well-known techniques for handing discrete optimization variables. We point the reader to [22, 23] 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 the original objective function, $J(W, H, \{y_{ij}\}_{ij \in \bar{S}})$, with respect to y_{ij}, optimize the expected value of the objective function 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. The solution of an easier optimization problem is used as the starting point of a harder optimization. Homotopy refers to this kind of smooth deformation of objectives.

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

$$\begin{aligned} \underset{W \geq 0, H \geq 0, \{p_{ij}\}_{(i,j) \in \bar{S}}}{\arg\max} & J_T(W, H, \{p_{ij}\}_{ij \in \bar{S}}) = \\ & \lambda \|W\|_F^2 + \gamma \|H\|_F^2 + \sum_{(i,j) \in S} C_{ij} V(1, w_i^T h_j) \\ + \sum_{(i,j) \in \bar{S}} & C_{ij} \left(p_{ij} V(1, w_i^T h_j) + (1 - p_{ij}) V(0, w_i^T h_j) \right) \\ & -T \sum_{(i,j) \in \bar{S}} H(p_{ij}) \quad \text{subject to: } \frac{1}{|\bar{S}|} \sum_{ij} p_{ij} = r \end{aligned} \quad (7)$$

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, while keeping p_{ij}'s fixed. This is a call to a stand-alone weighted NMF solver as we show below. 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. We outline these steps in more detail for the squared loss, i.e., $V(t, y) = \frac{1}{2}(t - y)^2$.

4.1 Optimizing W, H for fixed p_{ij} variables

For fixed p, the optimization over W, H involves the first four terms of Equation 7. This sub-problem can be re-written as,

$$\begin{aligned} \underset{W,H}{\arg\min}\ & \lambda \|W\|_F^2 + \gamma \|H\|_F^2 + \|S \otimes (X - WH)\|_F^2 + \\ & \sum_{ij} C_{ij} \left[p_{ij} \left(1 - w_i^T h_j\right)^2 + (1 - p_{ij}) (w_i^T h_j)^2 \right] \end{aligned} \quad (8)$$

where $\otimes$ denotes elementwise product and the matrix S is given by $S_{ij} = \sqrt{C_{ij}}$. With some simple re-arrangements, it is easy to see that the solutions to the above problem can be obtained by solving,

$$\underset{W,H}{\arg\min}\ \lambda \|W\|_F^2 + \gamma \|H\|_F^2 + \|S \otimes (U - WH)\|_F^2 \quad (9)$$

where $U = X + P$, P being a matrix whose elements equal p_{ij} when $(i, j) \in \bar{S}$ and 0 when $(i, j) \in S$. Thus, the sub-problem of minimizing W, H for fixed p_{ij}'s is the weighted NMF problem of Equation 9. The solution can be obtained by alternating between the following two multiplicative update steps,

$$H = H \otimes \frac{W^T (S \otimes U)}{W^T (S \otimes (WH)) + \gamma H} \quad (10)$$

$$W = W \otimes \frac{(S \otimes U) H^T}{(S \otimes (WH)) H^T + \lambda W} \quad (11)$$

where division is elementwise. The iterations are started from the previous values of W, H maintained in an outer loop, or from random matrices or other baseline solutions if the first outer-loop is being executed. Once inside this sub-routine, steps in Equations 10 and 11 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. With simple modifications to the update steps above, we can handle other loss functions such as the generalized KL-divergence and also other regularizers such as the l_1 norm of W and H.

4.2 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 Equation 7, subject to the balance constraint. Let ν be the Lagrange multiplier corresponding to the balance constraint, $\frac{1}{|\bar{S}|}\sum_{ij} p_{ij} = r$. Define,

$$g_{ij} = S_{ij}\left[V(1, o_{ij}) - V(0, o_{ij})\right] \tag{12}$$

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

$$p_{ij} = \frac{1}{1 + e^{\frac{g_{ij}-\nu}{T}}} \tag{13}$$

where ν can be found by substituting the above in the balance constraint and solving a one-dimensional non-linear equation in ν,

$$\frac{1}{|\bar{S}|}\sum_{ij\in\bar{S}} \frac{1}{1 + e^{\frac{g_{ij}-\nu}{T}}} = r \tag{14}$$

The details of solving this kind of one-dimensional non-linear equation is given in [22].

4.3 Outer Loop & Final Recommendations

For a fixed value of T, we optimize W, H and the p_{ij} variables using the alternating optimization scheme outlined above. This involves repeatedly executing Equations 10, 11, 13 until some termination criterion is satisfied. In Section 5, we report the performance of $ZaOV$ at a fixed value of T and the study the sensitivity to this choice with respect to recommendation quality.

We are currently studying the behavior of full annealing optimization (not reported in this paper). In this case, $ZaOV$ is initially started from a high value of T. Then, the entropy of p_{ij} dominates the objective function and the associated maximum entropy solution under balance constraints is obtained by setting $p_{ij} = r$. The W, H optimization then essentially solves a soft ZAN problem where zeros are replaced by r. As T is reduced at a rate specified by an annealing rate parameter, p_{ij} variables begin to harden (i.e., be close to 0 or 1) and behave as discrete variables, but their optimization is initialized from the previous solution obtained at higher value of T.

The gradual reduction of T may be seen as solving a sequence of optimization problems that slowly shrinks the effect of an entropy based regularizer parameterized by T. The inner fixed-T optimization can be stopped based on KL divergence between successive values of p_{ij}, while the outer optimization can be stopped based on net entropy of p_{ij} variables, as suggested in [23]. We outline the detailed steps in the table marked Algorithm 1. Once the factor matrices are learnt, the data matrix is reconstructed and thresholded to extract (client, product) recommendations (see the table marked Algorithm 2). In Section 5, we study the performance of the algorithm in terms of precision-recall curves generated by varying the recommendation threshold.

Algorithm 1 ZaOV

Input: *data matrix:* $X \in \mathbf{R}^{m\times n}$, *cost matrix:* $C \in \mathbf{R}^{m\times n}$
positive class ratio: r
regularization parameters: λ, γ, T
convergence parameters: $\tau, \epsilon_{in}, \epsilon_{out}, iter_{out}, iter_{in}, iter_{nmf}$

Output: W, H that minimize Equation 7

▷ *Intialize* $S, P, AnnealingRate, W, H$
Set $S : S_{ij} = \sqrt{C_{ij}}$
Set P: $P_{ij} = r$ *if* $(i,j) \in \bar{S}$ *else* $P_{ij} = 0$
$AnnealingRate = 1.5$

for *cycle* $= 1$ *to* $iter_{out}$ **do**
  for $i = 1$ *to* $iter_{in}$ **do**

    $U = X + P$

    ▷ *Update* H *and* W *keeping all other variables fixed:*
    $W_0 = W, H_0 = H$
    for $t = 1$ *to* $iter_{nmf}$ **do**
      Update H^t using Equation 10
      Update W^t using Equation 11
      ▷ *Monitor objective function for convergence*
      Update $J_T^t = J_T(W^t, H^t, P^i)$
      break if
        $J_T^t - J_T^{t-1} < \tau J_T^{t-1}$
    end for
    $W = W^t, H = H^t$

    Set $P_{old} = P$ ▷ *To monitor convergence*
    Update P by using Equations 12, 14 and then 13
    break if
      $KL_{divergence}(P, P_{old}) > \epsilon_{kl}$
  end for

  ▷ *Update T:*
    $T = T/AnnealingRate$
  break if
    $Entropy(P) < \epsilon_{out}$
end for

Remarks: We close this section with some remarks on variations of the current theme. Firstly, it may be possible to re-arrange the alternating optimization so that p_{ij} variables are re-optimized after each W and H update. It would be interesting to study the effect of such a re-arrangement. Secondly, while we impose the notion of annealing and smooth deformation of the objective function with respect to p_{ij} variables alone, similar ideas can be used for W, H also. One way to do this is to gradually reduce λ, γ starting from large values to their final values. In this way, the degree of non-convexity of the original objective function can potentially be controlled along multiple dimensions in the annealing process. Finally, for large-scale problems, we may want to avoid explicitly optimizing all (i, j) variables in $\bar{S}$, but rather work with a smaller subset. Such a subset can be randomly chosen. The question of optimal choices for this subset leads to interesting new technical problems. Another direction is that of choosing the regularization parameters and the positive class ratio in a principled manner using procedures like cross-validation.

Algorithm 2 Extracting Recommendations

Input: *Factor Matrices:* $W \in \mathbf{R}^{m\times k}, H \in \mathbf{R}^{k\times n}$
threshold

Output: *Recommendation matrix:* $\mathbf{Z} \in \mathbf{R}^{m\times n}$

$\triangleright$ *Reconstruct the data matrix*
$\hat{X} = WH$
$Z = \hat{X} \otimes (\hat{X} \geq threshold)$ $\triangleright$ *elements in* $\hat{X}$ *below threshold are set to* 0.

$\triangleright$ *Recommend product (movie)* j *to client* i *if the following holds:* $Z_{ij} = 1$ *but* $X_{ij} = 0$

5. EMPIRICAL STUDY

5.1 Dataset

We conducted experiments on the MovieLens dataset available at: `http://www.grouplens.org/system/files/ml-data_0.zip`. The data consists of 100,000 ratings on an integer scale from 1 to 5 given to 1642 movies by 943 users. To simulate one-class experiments, we removed all 3 and below ratings, and relabeled ratings 4 and 5 as 1, to then obtain a binary customer-movie matrix. We created random training-test splits of positive customer-movie pairs in the ratio 60%-to-40% respectively. All results reported in this section are averaged over 10 random splits.

5.2 Evaluation

For the purposes of evaluation, we computed precision-recall curves in the following manner. After computing W, H from $ZaOV$, we extract recommendations from Algorithm 2 at a pre-defined threshold. At that threshold, we compute the following quantities:

$$Precision = \frac{\#TestSetHits}{\#Recommendations}$$

$$Recall = \frac{\#TestSetHits}{\#TestSetPositives}$$

where $\#Recommendations$ is the number of user-movie recommendations made and $\#TestSetHits$ is the subset of recommended user-movie pairs that are one-valued in the test set (these numbers depend on the recommendation threshold); $\#TestSetPositives$ is the number of ones in the test set. We report the Area under the Precision-Recall Curves (abbreviated as AUC) generated as the recommendation threshold is varied.

5.3 Baselines and Competing Techniques

We compared our one class approach with the the following baselines. We mentioned the first two earlier in the paper as routine one-class approaches. The last three are schemes that were proposed in [20, 19] and implemented with unconstrained factorizations; here, we apply them with NMF.

1. ZAM: treating *zeros as missing* i.e., solving Equation 1 with $C_{ij} = 1, 1 \leq i \leq m, 1 \leq j \leq n$.
2. ZAN: treating *zeros as negative* i.e., solving Equation 2 with $C_{ij} = 1, 1 \leq i \leq m, 1 \leq j \leq n$.
3. wZAN (unif): a weighted version of ZAN where zeros are treated as negative, but a uniform weight with value less than 1 is additionally imposed (i.e., $C_{ij} < 1, 1 \leq i \leq m, 1 \leq j \leq n$, in the last term of Eqn. 2). This weighting implements the intuition that the confidence of unknown labels being negative is lower than the confidence of positive labels. In particular, we report the best performance over the following weights
$$\delta_k = \frac{n_+}{2^k n_0} \quad (15)$$
where n_+ are the number of positive labels, and n_0 are the number of zero-entries in X, with $0 \leq k \leq 5$.
4. wZAN (item-oriented): Here, the weights are not uniform and the j^{th} *item / column* has its own weight proportional to $m - \sum_i X_{ij}$. [20] considered user-item matrices and therefore refer to this column weighting as item-oriented. In the sales recommendation setting, this would be a product oriented weighting scheme, and for movie recommendation, it would impose movie-specific weights. In particular, we report the best performance over the weights $\forall j : C_{ij} = \delta_k(m - \sum_i X_{ij}), k = 0, 1, 2, 3$ where δ_k is the same as defined in Equation 15 for wZAN (unif). This weighting scheme implements the intuition that if an item has less positive examples, the missing data for this item is negative with higher probability.
5. wZAN (user-oriented): Here, the weights are not uniform and the i^{th} *user/ row* has its own weight proportional to $\sum_j X_{ij}$. In the sales recommendation setting, this would be a client-oriented weighting scheme, and for movie recommendation, it would impose user-specific weights. In particular, we report the best performance over the following weights $\forall j : C_{ij} = \delta_k \sum_i X_{ij}, k = 0, 1, 2, 3$ where δ_k is the same as defined for wZAN (unif). This weighting scheme implements the intuition that a user has more positive examples, it is more likely that the other items are less preferred, that is, the missing data for this user/client is negative with higher probability.
6. Popularity: In this scheme, each client-item pair is ranked based on the popularity of the item (number of clients who purchased it) and the purchase volume of the client (number of products bought). Then a client-product pair (i, j) is ranked as follows,
$$rank(i, j) = (\#clients)[rank(i) - 1] + rank(j)$$
Recommending based on this ranking in effect attempts to sell products (in the order of their popularity) to clients (in the order of their loyalty).

5.4 Results

In Table 1, we report the Area under the precision recall curve for each of the 6 baselines, and compare them with $ZaOV$ 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. For $wZAN$ with uniform, item-oriented and user-oriented weightings, we optimized the value of δ_k in Equation 15 over 7 choices of $k, 0 \leq k \leq 6$.

As expected, since it only uses a small set of positive examples, ZAM returns the worst performance. It is outperformed by the simple popularity based heuristic. ZAN performs substantially better than ZAM and Popularity-based

Table 1: Comparison of all methods in terms of Area under Precision-Recall Curve

Methods	$rank = 5$	$rank = 10$	$rank = 15$
ZAM	1.10 ± 0.09	1.42 ± 0.14	1.44 ± 0.19
Popularity	2.29 ± 0.11	2.29 ± 0.11	2.29 ± 0.11
ZAN	25.09±0.24	26.05±0.33	24.37±0.28
wZAN (uniform)	24.38±0.37	25.47±0.47	24.03±0.49
wZAN (item based)	25.45±0.63	27.17±0.60	26.39±0.59
wZAN (user based)	16.36±0.22	16.98±0.34	17.05±0.46
ZoAV (proposed)	**26.33**±0.65	**28.09**±0.71	**27.36** ±0.54

schemes. The performance of ZAN becomes worse with user-oriented weighting, but improves with item-oriented weighting – the opposite observation was made in [20] in the context of news recommendation. Finally, initializing from the best baseline solution (item-based $wZAN$), optimization using $ZaOV$ further leads to statistically significant performance improvements.

For $ZoAV$ we used the following internal optimization parameters: $iter_{in} = 40, iter_{out} = 1, iter_{nmf} = 100, \epsilon_{out} = 10^{-8}, \epsilon_{in} = 10^{-6}, \tau = 0.0001$. The following hyperparameters were used: $T = 30$, $r = 0.1$. Keeping T fixed at 30, we report performance sensitivity to r in Table 2. Similarly, keeping r fixed at 0.1, in Table 3 we report performance sensitity with respect to choice of T. We see that $ZaOV$ tends to be robust to the selection of r, as the performance is stable for $r \in [0.05, 0.3]$. Large values of r bias the algorithm towards filling up large portions of the missing values with ones, which does not suit the sparse nature of collaborative filtering problems. With regards to sensitivity with respect to choice of T, we see that $ZaOV$ returns similar performance for various values of T.

We are currently studying the behavior of $ZaOV$ under annealing, i.e., as T is gradually reduced.

Table 2: Sensitivity to r

r	0.01	0.05	0.1	0.2	0.3	0.7
AUC	0.2788	0.2825	0.2859	0.2849	0.2819	0.1528

Table 3: Sensitivity to T

T	10	50	100	1000	10000
AUC	0.2878	0.2891	0.2867	0.2887	0.2881

6. CONCLUDING COMMENTS

In this paper, we have sketched a principled, novel optimization approach to one-class collaborative filtering. We have drawn on non-convex optimization techniques previously utilized in the context of transductive SVMs for semi-supervised learning. Our method jointly learns a non-negative matrix factorization model for collaborative filtering while optimizing for unknown discrete label variables. 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. 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. Uninterpretable recommendations are more likely to be dismissed as not being well-tailored to the needs of the client or the specifications of the product, as recognized by the sellers and marketers. In this respect, latent factor models are relatively weak as compared to rule-oriented recommender systems based on frequent itemset or sub-sequence mining. An open research direction is how to extract interpretability from latent factors and provide meaningful explanations for why a client ought to be sold a recommended product. Also, business evaluation of a recommender system is somewhat different from measures like precision and recall. To add real business value recommender systems need to go beyond the current practices of the sales teams, and generate high win-probability recommendations that are also somewhat non-obvious. Finally, practical considerations such as expected revenue and time taken to close the deal are also important business factors in judging the value of a recommendation.

7. REFERENCES

[1] R. Bell, J. Bennett, Y. Koren, and C. Volinsky. The million dollar programming prize. In *IEEE Spectrum*, 2009.

[2] V. D. Blondel, N.-D. Ho, and P. V. Dooren. Weighted non-negative matrix factorization and face feature extraction. In *Image and Vision Computing*, 2008.

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

[4] O. Chapelle, V. Sindhwani, and S. Keerthi. Optimization techniques for semi-supervised support vector machines. *Journal of Machine Learning Research*, 9:203–233, 2006.

[5] A. Cichocki, R. Zdunek, A. Phan, and S. Amari. *Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis and Blind Source Separation*. Wiley, 2009.

[6] C. Ding, T. Li, and W. Peng. On the equivalence between non-negative matrix factorization and probabilistic latent semantic indexing. *Computational Statistics and Data Analysis*, pages 3913–3927, 2008.

[7] D. G. et. al. Using collaborative filtering to weave an information tapestry. In *Comm. ACM, 35*, 1992.

[8] 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*, volume 1, pages 49–75, 2000.

[9] T. Hofmann. Probabilistic latent semantic indexing. *Proceeding of SIGIR*, pages 50–57, 1999.

[10] T. Hofmann. Latent semantic analysis for collaborative filtering. In *ACM Transactions on Information Systems*, 2004.

[11] P. Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning Research*, 5:1457–1469, 2004.

[12] T. E. J.-P. V. J. Abernethy, F. Bach. A new approach to collaborative filtering: Operator estimation with spectral regularization. In *Journal of Machine Learning Research*, volume 10, pages 803–826, 2009.

[13] T. Joachims. Transductive inference for text classification using support vector machines. In *ICML*, 1999.

[14] Y. Koren. Collaborative filtering with temporal dynamics. In *KDD*, 2009.

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

[16] C. Lin. Projected gradient methods for non-negative matrix factorization. *Neural Computation*, 19:2756–2779, 2007.

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

[18] 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.

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

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

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

[22] V. Sindhwani and S. Keerthi. Large scale semi-supervised linear svms. In *SIGIR*, 2006. http://vikas.sindhwani.org/semisup_techreport06.ps.

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

[24] N. Srebro and T. Jaakkola. Weighted low-rank approximations. In *ICML*, 2003.

[25] R. B. Yehuda Koren and C. Volinsky. Matrix factorization techniques for recommender systems. In *IEEE Computer*, volume 42 (8), pages 30–37.