Content-Boosted Matrix Factorization Techniques for Recommender Systems

Jennifer Nguyen[1] and Mu Zhu[2*]

[1]*Computer Science Department, University College London, Gower Street, London, England WC1E 6BT*

[2]*Department of Statistics and Actuarial Science, University of Waterloo, 200 University Avenue West, Waterloo, Ontario N2L 3G1, Canada*

Received 14 March 2012; revised 20 December 2012; accepted 20 December 2012
DOI:10.1002/sam.11184
Published online 2 April 2013 in Wiley Online Library (wileyonlinelibrary.com).

Abstract: Many businesses are using recommender systems for marketing outreach. Recommendation algorithms can be either based on content or driven by collaborative filtering. We study different ways to incorporate content information directly into the matrix factorization approach of collaborative filtering. These content-boosted matrix factorization algorithms not only improve recommendation accuracy, but also provide useful insights about the contents, as well as make recommendations more easily interpretable. © 2013 Wiley Periodicals, Inc. Statistical Analysis and Data Mining 6: 286–301, 2013

Keywords: collaborative filtering; regression; shrinkage; SVD

1. INTRODUCTION

Many businesses today are using the Internet to promote and sell their products and services. Through the Internet, businesses can easily market many items to a large number of consumers. With a vast number of items, however, consumers may be overwhelmed by their choices. That is why, in an effort to maintain customer satisfaction and loyalty, many businesses have also integrated the use of recommender systems in their marketing strategies. For example, the online store www.amazon.com will suggest, based on a user's past purchases, products that he/she may be interested in.

Recommender systems today typically use one of two approaches: the *content-based* approach, or the *collaborative filtering* (CF) approach. In the content-based approach (e.g., Pandora, www.pandora.com), a profile is created for each user and for each item. The user profile describes the contents that he/she likes, and the item profile describes the contents that it contains. To a given user, the system recommends items that match his/her profile. In the CF approach (e.g. Netflix, www.netflix.com), users who have rated the same items closely are considered to have similar preferences overall. To a given user, the system recommends items that similar users have rated favorably before.

For an extensive review and discussion of different CF algorithms as well as an up-to-date and comprehensive bibliography, we refer the readers to a recent article by Feuerverger et al. [1]. While various algorithms have been adapted for the recommendation problem including restricted Boltzmann machines [2], most CF algorithms can be classified into two broad categories [3]: those based on *nearest neighbors* and those based on *matrix factorization*. While the nearest-neighbor approach is more intuitive, the matrix-factorization approach has gained popularity as a result of the Netflix contest [4].

1.1. Focus of Paper

Perhaps the most important lessons from the Netflix contest are that, in terms of prediction accuracy, it is often difficult for any single algorithm to outperform an ensemble of many different algorithms [1,4], and that algorithms of different flavors, when taken alone, often have similar predictive power for a given problem, even though they may each capture different effects.

For example, as shown by Feuerverger et al. [1] in their Table 1, on the Netflix data, a neighborhood-based method alone (labeled 'kNN' in their table) had a root

* *Correspondence to:* Mu Zhu (m3zhu@uwaterloo.ca)

mean squared error (RMSE) of 0.9174 whereas a method based on matrix factorization (MF) alone (labeled 'SVD' in their table) had an RMSE of 0.9167—very close indeed. A significant drop in the RMSE (to 0.8982) was achievable only when the two classes of methods were combined together; see also ref. [3]. And it is widely known that the ultimate winner in the Netflix contest (with an RMSE of 0.8572) was an ensemble of no fewer than 800 different algorithms.

Therefore, a research project on CF can either focus on new classes of CF algorithms that are fundamentally different from existing ones, or focus on improvements or extensions *within* a certain class. For projects of the first type, the key question is whether the new class of algorithms is better than other classes. For those of the second type, the key question is whether the proposed extension adds any value when compared with baseline algorithms in the *same* class. The research we will report in this paper is strictly of the second type. In particular, we focus on the MF approach only.

1.2. The 'Cold Start' Problem

One advantage of the CF approach is that it does not require extra information on the users or the items; thus, it is capable of recommending an item without understanding the item itself [5]. However, this very advantage is also the root cause of the so-called 'cold start' problem, which refers to the general difficulty in performing CF for users and items that are relatively new. By definition, newer users are those who have not rated many items, so it is difficult to find other users with similar preferences. Similarly, newer items are those which have not been rated by many users, so it is difficult to recommend them to anyone.

Various ideas have been proposed to deal with the 'cold start' problem. Park et al. [6] suggested using so-called 'filterbots'—artificial items or users inserted into the system with pre-defined characteristics. For instance, an action-movie filterbot can make recommendations to new users who have only liked one or two action movies. More recently, Zhao et al. [7] suggested *shared* CF, an ensemble technique that aggregates predictions from several different systems. Since one recommender system may have data on user-item pairs that another one does not, it is possible to improve recommendations by sharing information across different systems.

Another common approach for dealing with the 'cold start' problem is to fill in the missing ratings with 'pseudo' ratings before applying CF. For example, Goldberg et al. [8] did this with principal component analysis; Nguyen et al. [9] did this with rule-based induction; while Melville et al. [10] did this with a hybrid, two-step approach, creating 'pseudo' ratings with a content-based classifier.

1.3. Objectives and Contributions

The key idea behind the hybrid approach is to leverage supplemental information [11]. Many recent works have taken this basic idea to new heights, successfully exploiting supplemental information from different sources and in various forms, for example, tagging history [12], personality traits [13,14], social networks [15,16], and Wikipedia articles [17].

In this paper, we focus on a particular type of supplemental information—content information about the individual items. For example, for recipes [18] we may know their ingredient lists; for movies [4] we may know their genres. Moreover, we focus on ways to take advantage of such content information *directly* in the matrix factorization approach, not by using a hybrid or two-step algorithm. We refer to our suite of algorithms as 'content-boosted MF algorithms'.

Not only can these content-boosted algorithms achieve improved recommendation accuracy (see Section 4.8), they can also produce more interpretable recommendations (see Section 5.1), as well as furnish useful insights about the contents themselves that are otherwise unavailable (see Section 5.2). More interpretable recommendations are becoming ever more desirable commercially, because users are more likely to act on a recommendation if they understand why it is being made to them [19,20], while better understandings of contents can facilitate the creation of new products, such as recipes with substitute ingredients.

One of the big lessons from the Netflix contest was the need to develop 'diverse models' that capture 'distinct effects, even if they are very small effects' [21]. Since most methods do not take content information into account, the value of our content-boosted algorithms is clear, and we fully expect that our algorithms will further enhance any existing ensembles.

1.4. Outline

We proceed as follows. In Section 2, we give a brief review of the MF approach for collaborative filtering. In Section 3, we present a number of different content-boosted MF algorithms. In Section 4, we describe the data sets we used and the experiments we performed to study and evaluate various algorithms. In Section 5, we describe useful by-products from these content-boosted MF techniques. In Section 6, we briefly mention some open problems for future research. We end in Section 7 with a brief summary.

2. MF: A BRIEF REVIEW

Before we start, it is necessary to review the basic MF method briefly. Our review follows the work of Koren et al. [4].

2.1. Notation

Given a set of users $U = \{u_1, \ldots, u_N\}$, and a set of items $I = \{i_1, \ldots, i_M\}$, let r_{ui} denote the rating given by user u to item i. These ratings form a user–item rating matrix, $\mathbf{R} = [r_{ui}]_{N\times M}$. In principle, r_{ui} can take on any real value but, in practice, r_{ui} is typically binary, indicating 'like' and 'dislike', or integer-valued in a certain range, indicating different levels of preferences, e.g., $r_{ui} \in \{1, \ldots, 5\}$.

Often, the rating matrix $\mathbf{R}$ is highly sparse with many unknown entries, as users typically are only able to rate a small fraction of the items—recall the 'cold start' problem discussed briefly in Section 1.2. We denote

$$T = \{(u, i) : r_{ui} \text{ is known}\}$$

as the set of indices for known ratings. Given an unknown (user, item)-pair, $(u, i) \notin T$, the goal of the recommender system is to predict the rating that user u would give to item i, which we denote by $\widehat{r}_{ui}$. Furthermore, we define

$$T_{u\cdot} \equiv \{i : (u, i) \in T\}$$

to be the set of items that have been rated by user u, and

$$T_{\cdot i} \equiv \{u : (u, i) \in T\}$$

to be the set of users who have rated item i.

2.2. Normalization by ANOVA

Despite its overwhelming simplicity, an ANOVA-type of model often captures a fair amount of information in the rating data [1,4]. The simplest ANOVA-type model used in the literature consists of just main effects, i.e.,

$$r_{ui} = \mu + \alpha_u + \beta_i + \epsilon_{ui}, \tag{1}$$

where ϵ_{ui} is white noise, μ is the overall mean, α_u represents a user-effect, and β_i represents an item-effect. These two main effects capture the obvious fact that some items are simply better liked than others, while some users are simply more difficult to please.

It is common in the literature to normalize the rating matrix $\mathbf{R}$ by removing such an ANOVA-type model before applying any matrix-factorization (or nearest-neighbor) methods (see ref. [4]). In all of our experiments reported below, we followed this common practice, that is, all matrix-factorization algorithms were applied to $r_{ui} - \widehat{\mu} - \widehat{\alpha}_u - \widehat{\beta}_i$, and the predicted rating was actually $\widehat{r}_{ui} + \widehat{\mu} + \widehat{\alpha}_u + \widehat{\beta}_i$, where $\widehat{r}_{ui}$ was the prediction from the matrix-factorization algorithm, and $\widehat{\mu}$, $\widehat{\alpha}_u$, $\widehat{\beta}_i$ were the MLEs of μ, α_u, β_i. In order not to further complicate our notation, however, this detail will be suppressed in our presentation, and we still use the notations, r_{ui} and $\mathbf{R}$, despite the normalization step.

2.3. Matrix Factorization

To predict unknown ratings in $\mathbf{R}$, the MF approach uses all the known ratings to decompose the matrix $\mathbf{R}$ into the product of two low-rank, latent feature matrices, one for the users, $\mathbf{P}_{N\times K}$, and another for the items, $\mathbf{Q}_{M\times K}$, so that

$$\mathbf{R} \approx \widehat{\mathbf{R}} = \mathbf{P}\mathbf{Q}^{\mathrm{T}} = \underbrace{\begin{bmatrix} \mathbf{p}_1^{\mathrm{T}} \\ \mathbf{p}_2^{\mathrm{T}} \\ \vdots \\ \mathbf{p}_N^{\mathrm{T}} \end{bmatrix}}_{N\times K} \underbrace{\begin{bmatrix} \mathbf{q}_1 & \mathbf{q}_2 & \cdots & \mathbf{q}_M \end{bmatrix}}_{K\times M}. \tag{2}$$

The latent feature vectors—$\mathbf{p}_u$ for user u ($u = 1, 2, ..., N$) and $\mathbf{q}_i$ for item i ($i = 1, 2, ..., M$)—are K-dimensional, where $K \ll \min\{M, N\}$ is pre-specified. The predicted rating for the user-item pair (u, i) is simply

$$\widehat{r}_{ui} = \mathbf{p}_u^{\mathrm{T}}\mathbf{q}_i.$$

Intuitively, one can imagine a K-dimensional map, in which $\mathbf{p}_u$ and $\mathbf{q}_i$ are the (latent) coordinates for user u and item i, respectively, and all the information that we need in order to make recommendations is contained in such a map—users will generally like items that are nearby. Latent-coordinate models have a long history, e.g., principal component analysis, factor analysis, multidimensional scaling, and so on (see ref. [22]).

Mathematically, the factorization in Eq. (2) can be achieved by solving the optimization problem,

$$\min_{\mathbf{P},\mathbf{Q}} \quad \|\mathbf{R} - \mathbf{P}\mathbf{Q}^{\mathrm{T}}\|^2, \tag{3}$$

where $\|\cdot\|$ is the Frobenius norm. To prevent over-fitting, it is common to include a regularization penalty on the sizes of $\mathbf{P}$ and $\mathbf{Q}$, turning the optimization problem above into

$$\min_{\mathbf{P},\mathbf{Q}} \quad \|\mathbf{R} - \mathbf{P}\mathbf{Q}^{\mathrm{T}}\|^2 + \lambda\left(\|\mathbf{P}\|^2 + \|\mathbf{Q}\|^2\right). \tag{4}$$

From a Bayesian point of view, the first part of the objective function in Eq. (4) can be viewed as coming from a Gaussian likelihood function; the regularization penalties

can be viewed as coming from spherical Gaussian priors on the user and item feature vectors; and the solution to the optimization problem itself is then the so-called maximum *a posteriori* estimate [23].

2.4. Relative Scaling of Penalty Terms

Feuerverger et al. [1] used empirical Bayes analysis to argue that one should, in principle, always penalize $\|\mathbf{p}_u\|^2$ and $\|\mathbf{q}_i\|^2$ by different amounts. In practice, their advice is not followed always because the extra computational burden to select two tuning parameters rather than one is substantial, and the resulting payoff in terms of performance improvement may not be significant.

In our work, we found it convenient to scale the second penalty term—the one on $\|\mathbf{Q}\|^2$—by a factor $\gamma > 0$ such that, regardless of how many users (N) and how many items (M) there are, the penalty on $\|\mathbf{Q}\|^2$ is always on the same order of magnitude as the penalty on $\|\mathbf{P}\|^2$. We will come back to this point later (see Section 4.1).

Furthermore, since most entries in $\mathbf{R}$ are unknown, we can only evaluate the first term in Eq. (4) over known entries $(u, i) \in T$. This means the optimization problem actually solved in practice is:

$$\min_{\mathbf{P},\mathbf{Q}} L_{\text{BL}}(\mathbf{P}, \mathbf{Q}) = \sum_{(u,i)\in T} (r_{ui} - \mathbf{p}_u^{\text{T}}\mathbf{q}_i)^2 + \lambda \left(\sum_u \|\mathbf{p}_u\|^2 + \gamma \sum_i \|\mathbf{q}_i\|^2 \right). \quad (5)$$

The subscript 'BL' stands for 'baseline'. For the purpose of comparison, we will refer to this method below as the baseline matrix factorization method, or simply the baseline (BL) algorithm.

2.5. Alternating Gradient Descent

With both $\mathbf{P}$ and $\mathbf{Q}$ being unknown, the optimization problem (5) is not convex. It can be solved using an alternating gradient descent algorithm [4], moving along the gradient with respect to $\mathbf{p}_u$ while keeping $\mathbf{q}_i$ fixed, and vice versa.

Let ∇_u^{BL} denote the derivative of L_{BL} with respect to $\mathbf{p}_u$ and ∇_i^{BL}, its derivative with respect to $\mathbf{q}_i$. Then,

$$\nabla_u^{\text{BL}} \propto \sum_{i\in T_{u\cdot}} -(r_{ui} - \mathbf{p}_u^{\text{T}}\mathbf{q}_i)\mathbf{q}_i + \lambda\mathbf{p}_u, \quad (6)$$

$$\nabla_i^{\text{BL}} \propto \sum_{u\in T_{\cdot i}} -(r_{ui} - \mathbf{p}_u^{\text{T}}\mathbf{q}_i)\mathbf{p}_u + \lambda\gamma\mathbf{q}_i, \quad (7)$$

for every $u = 1, 2, ..., N$ and $i = 1, 2, ..., M$. At iteration $(j + 1)$, the updating equations for $\mathbf{p}_u$ and $\mathbf{q}_i$ are:

$$\mathbf{p}_u^{(j+1)} = \mathbf{p}_u^{(j)} - \eta\nabla_u^{\text{BL}}\left(\mathbf{p}_u^{(j)}, \mathbf{q}_i^{(j)}\right), \quad (8)$$

$$\mathbf{q}_i^{(j+1)} = \mathbf{q}_i^{(j)} - \eta\nabla_i^{\text{BL}}\left(\mathbf{p}_u^{(j)}, \mathbf{q}_i^{(j)}\right), \quad (9)$$

where η is the step size or learning rate. The algorithm is typically initialized with small random entries for $\mathbf{p}_u$ and $\mathbf{q}_i$, and iteratively updated over all $u = 1, \ldots, N$ and $i = 1, \ldots, M$ until convergence (see Algorithm 1). We will say more about initialization later (see Section 4.5).

2.6. SVD and Other MF Techniques

In the CF literature, the MF approach outlined above is often dubbed the 'singular value decomposition (SVD) approach' (see refs. [1,4,24]). Strictly speaking, this is a bit misleading. The SVD is perhaps the single most widely used MF technique in all of applied mathematics; it solves the following problem:

$$\begin{aligned} \min \quad & \|\mathbf{R} - \mathbf{P}_*\mathbf{D}_*\mathbf{Q}_*^{\text{T}}\|^2 \\ \text{s.t.} \quad & \mathbf{D}_* \text{ is diagonal with rank } K, \\ & \mathbf{P}_*^{\text{T}}\mathbf{P}_* = \mathbf{I} \quad \text{and} \quad \mathbf{Q}_*^{\text{T}}\mathbf{Q}_* = \mathbf{I}. \end{aligned} \quad (10)$$

Algorithm 1 Alternating Gradient Descent Algorithm for Optimizing L_{BL} — Eq. (5)

Input: $\mathbf{R} = [r_{ui}]_{N\times M}$, K
Output: $\mathbf{P}$, $\mathbf{Q}$
1: **initialize** $j \leftarrow 0$ and choose $\mathbf{P}^{(0)}, \mathbf{Q}^{(0)}$ (see Section 4.5)
2: **repeat**
3: **for all** $u = 1, \ldots, N$ and $i = 1, \ldots, M$ **do**
4: compute ∇_u^{BL} and ∇_i^{BL} using (6)-(7)
5: update $\mathbf{p}_u^{(j+1)}$ and $\mathbf{q}_i^{(j+1)}$ with (8)-(9)
6: **end for**
7: **until** $[L_{\text{BL}}(\mathbf{P}^{(j)}, \mathbf{Q}^{(j)}) - L_{\text{BL}}(\mathbf{P}^{(j+1)}, \mathbf{Q}^{(j+1)})]/L_{\text{BL}}(\mathbf{P}^{(j)}, \mathbf{Q}^{(j)}) < \varepsilon$
8: **return** $\mathbf{P}, \mathbf{Q}$

By letting $\mathbf{P} = \mathbf{P}_* \mathbf{D}^{1/2}$ and $\mathbf{Q} = \mathbf{Q}_* \mathbf{D}^{1/2}$, SVD would give us

$$\mathbf{R} \approx \mathbf{P}\mathbf{Q}^{\mathrm{T}} \tag{11}$$

such that $\mathbf{P}^{\mathrm{T}}\mathbf{P} = \mathbf{D}_*^{1/2}\mathbf{P}_*^{\mathrm{T}}\mathbf{P}_*\mathbf{D}_*^{1/2} = \mathbf{D}_*$ is diagonal, meaning that $\mathbf{P}$ is an orthogonal matrix, and likewise for $\mathbf{Q}$. However, the MF approach outlined above does *not* require either $\mathbf{P}$ or $\mathbf{Q}$ to be orthogonal. To be sure, we confirmed this directly with the winners of the Netflix contest [25], who used this technique pervasively in their work. Without the orthogonality constraints, this would certainly raise identifiability and degeneracy questions for the optimization problem in Eq. (5), but these problems can be avoided *in practice* by carefully initializing the alternating gradient descent algorithm—we elaborate this in detail in Section 4.5 below.

Lee and Seung [26] popularized another MF technique called the non-negative matrix factorization (NMF), which is Eq. (3) with the additional non-negativity constraints that

$$P_{uk} \geq 0 \quad \text{and} \quad Q_{ik} \geq 0 \quad \text{for all} \quad u, i, k.$$

The NMF has been used to analyze a wide variety of data such as images [26] and gene expressions [27] to reveal interesting underlying structure. In recent years, it has also been used to perform CF (see refs. [28,29]) even though finding underlying structures in the data is often not the primary goal for CF. MF with either orthogonality constraints (e.g., SVD) or non-negativity constraints (e.g., NMF) is more sound mathematically, since the problem is somewhat ill-defined without any constraints. However, we will still focus only on the unconstrained version outlined above (see Section 2.3) since it remains the most dominant in the CF community, owing partly to its wide use in the 3-year-long Netflix contest.

3. CONTENT-BOOSTED MF

Now, suppose that, for each item i, there is a content vector $\mathbf{a}_i = [a_{i1}, \ldots, a_{iD}]$ of D attributes. Stacking these vectors together gives an attribute matrix, $\mathbf{A} = [a_{id}]_{M \times D}$. For simplicity, we assume that all entries in $\mathbf{A}$ are binary, i.e., $a_{id} \in \{0, 1\}$, each indicating whether item i possesses attribute d. In what follows, we study and compare different ways of incorporating this type of content information *directly* into the MF approach. We present two classes of methods with slightly different flavors. One class uses extra penalties with selective shrinkage effects (see Section 3.1), and the other uses direct regression constraints (see Section 3.2).

3.1. Alignment-Biased Factorization

To incorporate $\mathbf{A}$ into the MF approach, one idea is as follows: if two items i and i' share at least c attributes in common—call this the 'common attributes' condition, then it makes intuitive sense to require that their feature vectors, $\mathbf{q}_i$ and $\mathbf{q}_{i'}$, be 'close' in the latent space.

3.1.1. Details

For the MF approach, it is clear from Eq. (2) that the notion of closeness is modeled mathematically by the inner product in the latent feature space. Therefore, to say that $\mathbf{q}_i$ and $\mathbf{q}_{i'}$ are 'close' means that their inner product, $\mathbf{q}_i^{\mathrm{T}}\mathbf{q}_{i'}$, is large. We can incorporate this preference by adding another penalty, which we call the 'alignment penalty', to the optimization problem in Eq. (5).

For binary a_{id}, the 'common attributes' condition is easily expressed by $\mathbf{a}_i^{\mathrm{T}}\mathbf{a}_{i'} \geq c$. Let

$$\mathcal{S}_c(i) \equiv \{i' : \quad i' \neq i \quad \text{and} \quad \mathbf{a}_i^{\mathrm{T}}\mathbf{a}_{i'} \geq c\}.$$

We solve the following optimization problem:

$$\min_{\mathbf{P},\mathbf{Q}} L_{\mathrm{AB}}(\mathbf{P}, \mathbf{Q}) = L_{\mathrm{BL}}(\mathbf{P}, \mathbf{Q}) - \underbrace{\lambda\gamma \sum_{i=1}^{M} \sum_{i' \in \mathcal{S}_c(i)} \frac{\mathbf{q}_i^{\mathrm{T}}\mathbf{q}_{i'}}{|\mathcal{S}_c(i)|}}_{\text{alignment penalty}}, \tag{12}$$

where $L_{\mathrm{BL}}(\mathbf{P}, \mathbf{Q})$ is the baseline objective function given by Eq. (5), and the notation $|\mathcal{S}|$ means the size of the set $\mathcal{S}$. Notice that we make the alignment penalty adaptive to the size of $\mathcal{S}_c(i)$. The subscript 'AB' stands for 'alignment-biased'.

It is easy to see that the basic idea of alternating gradient descent still applies. For L_{AB}, the gradient with respect to $\mathbf{p}_u$ clearly remains the same, that is,

$$\nabla_u^{\mathrm{AB}} = \nabla_u^{\mathrm{BL}},$$

while the gradient with respect to $\mathbf{q}_i$ becomes

$$\nabla_i^{\mathrm{AB}} \propto \sum_{u \in T_{\cdot i}} -(r_{ui} - \mathbf{p}_u^{\mathrm{T}}\mathbf{q}_i)\mathbf{p}_u + \lambda\gamma \left[\mathbf{q}_i - \sum_{i' \in \mathcal{S}_c(i)} \frac{\mathbf{q}_{i'}}{|\mathcal{S}_c(i)|} \right]. \tag{13}$$

The updating equations are identical to Eqs. (8) and (9), except that ∇_u^{BL} and ∇_i^{BL} are replaced by ∇_u^{AB} and ∇_i^{AB}.

3.1.2. Differential shrinkage effects

The effect of the alignment penalty can be seen explicitly from Eq. (13) as shrinking the latent vector of each item

toward the centroid of items that share a certain number of attributes with it. This is the selective shrinkage effect that we alluded to earlier (see Section 3), and it plays a central role.

Next, we introduce a generalized/smoothed version of our alignment penalty (see Section 3.1.3) as well as a related but slightly different mathematical formulation (see Section 3.1.4). We will see that the main difference between these methods lies in their respective shrinkage effects—in each iteration, they shrink $\mathbf{q}_i$ towards slightly different centroids and by slightly different amounts; see the terms inside the square brackets in Eqs. (13), (16), (18), and (19).

3.1.3. A smooth generalization

An obvious generalization of the alignment penalty is to change Eq. (12) into

$$\min_{\mathbf{P},\mathbf{Q}} \quad L_{\text{gAB}}(\mathbf{P},\mathbf{Q}) = L_{\text{BL}}(\mathbf{P},\mathbf{Q}) - \underbrace{\lambda\gamma \sum_{i=1}^{M}\sum_{i'=1}^{M} w(i,i')\mathbf{q}_i^{\mathrm{T}}\mathbf{q}_{i'}}_{\text{gen. alignment penal.}}, \tag{14}$$

with

$$w(i,i') \propto \frac{\exp\left[\theta\left(\mathbf{a}_i^{\mathrm{T}}\mathbf{a}_{i'} - c\right)\right]}{1+\exp\left[\theta\left(\mathbf{a}_i^{\mathrm{T}}\mathbf{a}_{i'} - c\right)\right]}. \tag{15}$$

The 'proportional' relation '$\propto$' in Eq. (15) means the weights $w(i,i')$ are typically normalized to sum to unity, i.e., $\sum_{i'=1}^{M} w(i,i') = 1$ for any given i. The alignment penalty used in Eq. (12) corresponds almost everywhere to the special and extreme case of $\theta \to \infty$; for $\theta < \infty$, $w(i,i')$ is a smooth, monotonic function of the number of attributes shared by items i and i', rather than an abrupt, step function (see Fig. 1).

For L_{gAB} (14), the gradient with respect to $\mathbf{p}_u$ again remains the same, $\nabla_u^{\text{gAB}} = \nabla_u^{\text{AB}} = \nabla_u^{\text{BL}}$, while the gradient with respect to $\mathbf{q}_i$ simply becomes

$$\nabla_i^{\text{gAB}} \propto \sum_{u\in T_{\cdot i}} -(r_{ui} - \mathbf{p}_u^{\mathrm{T}}\mathbf{q}_i)\mathbf{p}_u + \lambda\gamma\left[\mathbf{q}_i - \sum_{i'=1}^{M} w(i,i')\mathbf{q}_i\right]. \tag{16}$$

Using smoother weights would allow all items that share attributes with i to contribute to the shrinkage effect, not just those that share at least a certain number of attributes with it. Moreover, their contributions would be adaptive—the amount of 'pull' that item i' exerts on the feature vector of item i is appropriated by how many attributes they share in common. Depending on how much information there is in the data, this could potentially enhance the effectiveness of the alignment penalty.

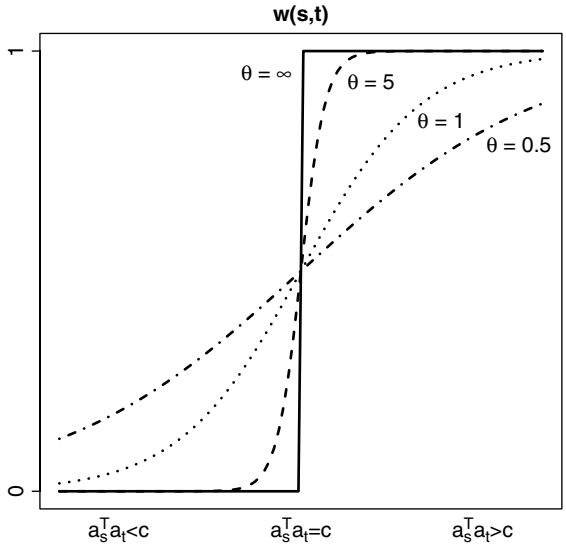

Fig. 1 The function $w(s,t)$ as given by Eq. (15), for $\theta = 0.5, 1, 5, \infty$.

3.1.4. A related method: tag informed CF

Noticing that many commercial recommender engines allow users to create personalized tags, Zhen et al. [12] proposed a method to exploit information from these tags. Following the work of Li and Yeung [30], their idea was to 'make two user-specific latent feature vectors as similar as possible if the two users have similar tagging history' by adding a tag-based penalty to the baseline optimization problem:

$$\min_{\mathbf{P},\mathbf{Q}} \quad L_{\text{BL}}(\mathbf{P},\mathbf{Q}) + \underbrace{\lambda \sum_{u=1}^{N}\sum_{u'=1}^{N} \|\mathbf{p}_u - \mathbf{p}_{u'}\|^2 w(u,u')}_{\text{tag-based penalty}},$$

where $w(u,u')$ is a measure of similarity between two users based on their tagging history. Interestingly, if we replace the word 'user' with 'item' and the phrase 'tagging history' with 'content' or 'attributes', the same idea can be applied to items, i.e.,

$$\min_{\mathbf{P},\mathbf{Q}} L_{\text{TG}}(\mathbf{P},\mathbf{Q}) = L_{\text{BL}}(\mathbf{P},\mathbf{Q}) + \lambda\gamma \sum_{i=1}^{M}\sum_{i'=1}^{M} \|\mathbf{q}_i - \mathbf{q}_{i'}\|^2 w(i,i'), \tag{17}$$

where $w(i,i')$ is the similarity between two items based on their content information, and the subscript 'TG' stands

for 'tag' indicating where the original idea came from. But since

$$\|\mathbf{q}_i - \mathbf{q}_{i'}\|^2 = \|\mathbf{q}_i\|^2 + \|\mathbf{q}_{i'}\|^2 - 2\mathbf{q}_i^{\mathrm{T}}\mathbf{q}_{i'},$$

it is easy to see that this leads to a similar but slightly different mathematical formulation, essentially consisting of

(1) penalizing $\|\mathbf{p}_u\|^2$ and $\|\mathbf{q}_i\|^2$ by different amounts (even if $\gamma = 1$) —in particular, the penalty in front of $\|\mathbf{q}_i\|^2$ is multiplied by $(1 + 2w_{i\cdot})$, where

$$w_{i\cdot} \equiv \sum_{i'=1}^{M} w(i, i');$$

and

(2) using the generalized version of our alignment penalty in Eq. (14), up to the specific choice of $w(i, i')$ itself.

Again, for L_{TG} in Eq. (17), the gradient with respect to $\mathbf{p}_u$ remains the same, $\nabla_u^{\mathrm{TG}} = \nabla_u^{\mathrm{AB}} = \nabla_u^{\mathrm{BL}}$, while the gradient with respect to $\mathbf{q}_i$ becomes

$$\nabla_i^{\mathrm{TG}} \propto \sum_{u \in T_{\cdot i}} -(r_{ui} - \mathbf{p}_u^{\mathrm{T}}\mathbf{q}_i)\mathbf{p}_u + \lambda\gamma \left[(1 + 2w_{i\cdot})\mathbf{q}_i - 2\sum_{i'=1}^{M} w(i, i')\mathbf{q}_{i'} \right]. \quad (18)$$

We can see that, when compared with Eq. (16), the selective shrinkage effect is somewhat attenuated in Eq. (18). This is most clearly seen if we normalize the weights to sum to one, i.e., $w_{i\cdot} = \sum_{i'=1}^{M} w(i, i') = 1$. Then, Eq. (18) simply becomes

$$\nabla_i^{\mathrm{TG}} \propto \sum_{u \in T_{\cdot i}} -(r_{ui} - \mathbf{p}_u^{\mathrm{T}}\mathbf{q}_i)\mathbf{p}_u + 3\lambda\gamma \left[\mathbf{q}_i - \frac{2}{3}\sum_{i'=1}^{M} w(i, i')\mathbf{q}_{i'} \right]. \quad (19)$$

Equation (19) reveals a curious factor of 2/3 in front of the weighted centroid, which clearly dampens this algorithm's corresponding shrinkage effect.

One of the similarity measures used by Zhen et al. [12] is the cosine similarity,

$$w(i, i') = \frac{\mathbf{a}_i^{\mathrm{T}}\mathbf{a}_{i'}}{\|\mathbf{a}_i\|\|\mathbf{a}_{i'}\|}. \quad (20)$$

Although other similarity measures can also be used, for binary attributes (see Section 3) the cosine similarity has an intuitive appeal as it amounts to something easily interpretable:

$$w(i, i') = \frac{(\#\text{ attributes shared by } i \text{ and } i')}{\sqrt{(\#\text{ attributes in } i)(\#\text{ attributes in } i')}}. \quad (21)$$

3.2. Regression-Constrained Factorization

Another idea for incorporating content information stored in the matrix $\mathbf{A}$ is to use a regression-style constraint, forcing each item feature vector to be a function of the item's content attributes, so that items with identical attributes are mapped to the same feature vector. This method was first introduced by our group in a short conference paper [18].

3.2.1. Details

Specifically, the constraint is

$$\mathbf{Q} = \mathbf{AB}, \quad (22)$$

where $\mathbf{B}$ is a $D \times K$ matrix. Each *column* of $\mathbf{B}$ behaves like a (vector) regression coefficient that maps the items to a latent feature using their content attributes. Each *row* of $\mathbf{B}$ can be viewed as a K-dimensional latent feature vector for the corresponding attribute.

Under the constraint in Eq. (22), the factorization in Eq. (2) becomes

$$\mathbf{R} \approx \mathbf{PQ}^{\mathrm{T}} = \mathbf{PB}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}} = \underbrace{\begin{bmatrix} \mathbf{p}_1^{\mathrm{T}} \\ \mathbf{p}_2^{\mathrm{T}} \\ \vdots \\ \mathbf{p}_N^{\mathrm{T}} \end{bmatrix}}_{N \times K} \underbrace{\mathbf{B}^{\mathrm{T}}}_{K \times D} \underbrace{\begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_M \end{bmatrix}}_{D \times M},$$

and the optimization problem (5) becomes

$$\min_{\mathbf{P},\mathbf{B}} L_{\mathrm{RC}}(\mathbf{P}, \mathbf{B}) = \sum_{(u,i) \in T} (r_{ui} - \mathbf{p}_u^{\mathrm{T}}\mathbf{B}^{\mathrm{T}}\mathbf{a}_i)^2 + \lambda\left(\sum_u \|\mathbf{p}_u\|^2 + \gamma\|\mathbf{B}\|^2 \right). \quad (23)$$

Again, the alternating gradient descent algorithm is applicable. The gradient of L_{RC} with respect to $\mathbf{p}_u$, ∇_u^{RC}, is the same as ∇_u^{BL}—Eq. (6), except that we replace $\mathbf{q}_i$ with $\mathbf{B}^{\mathrm{T}}\mathbf{a}_i$, i.e.,

$$\nabla_u^{\mathrm{RC}} \propto \sum_{i \in T_{u\cdot}} -(r_{ui} - \mathbf{p}_u^{\mathrm{T}}\mathbf{B}^{\mathrm{T}}\mathbf{a}_i)\mathbf{B}^{\mathrm{T}}\mathbf{a}_i + \lambda\mathbf{p}_u. \quad (24)$$

Using the fact that $d(\mathbf{x}^{\mathrm{T}}\mathbf{M}\mathbf{y})/d\mathbf{M} = \mathbf{x}\mathbf{y}^{\mathrm{T}}$, we can derive easily that the gradient of L_{RC} with respect to the matrix $\mathbf{B}$ is

$$\nabla_{\mathbf{B}}^{\mathrm{RC}} \propto \sum_{(u,i)\in T} -(r_{ui} - \mathbf{p}_u^{\mathrm{T}}\mathbf{B}^{\mathrm{T}}\mathbf{a}_i)\mathbf{a}_i\mathbf{p}_u^{\mathrm{T}} + \lambda\gamma\mathbf{B}. \quad (25)$$

At iteration $(j+1)$, the updating equations are:

$$\mathbf{p}_u^{(j+1)} = \mathbf{p}_u^{(j)} - \eta\nabla_u^{\mathrm{RC}}\left(\mathbf{p}_u^{(j)}, \mathbf{B}^{(j)}\right), \quad (26)$$

$$\mathbf{B}^{(j+1)} = \mathbf{B}^{(j)} - \eta\nabla_{\mathbf{B}}^{\mathrm{RC}}\left(\mathbf{p}_u^{(j)}, \mathbf{B}^{(j)}\right). \quad (27)$$

3.2.2. *Related literature*

The idea of incorporating regression relationships into latent factor models also has a long history. For example, ecologists used to apply a multivariate technique known as correspondence analysis [31,32] and fit so-called ordination models to sort species and geographical sites with latent coordinates (see ref. [33]); sites with similar conditions would have close-by coordinates, and likewise for species that prefer similar environments. Later, canonical correspondence analysis (CCA) was introduced [34], which constrains the latent site coordinates to be linear functions of actual environmental measurements at those sites. CCA has since become an extremely popular technique in the field of environmental ecology [35].

4. EXPERIMENTS

In this section, we describe the data sets we used and the experiments we performed to compare and evaluate various content-boosted MF algorithms against the baseline MF algorithm. We use the acronyms BL, AB, gAB, TG, and RC to refer to the algorithms; these acronyms should be self-evident from Sections 2 and 3. Table 1 briefly summarizes all the algorithms being compared and studied.

4.1. The Scaling Factor γ

As we have alluded to earlier (see Section 2.4), the purpose of the scaling factor γ is to balance the two penalties—the one on $\sum\|\mathbf{p}_u\|^2$ and the other on $\sum\|\mathbf{q}_i\|^2$ (or $\|\mathbf{B}\|^2$ in the case of RC)—so that the objective function is not dominated by either the user or the item side of the equation. Since the quantity $\sum\|\mathbf{p}_u\|^2$ remained constant in this paper and the algorithms differed only in terms of how they regularized the $\mathbf{q}_i$'s, the use of γ also allowed us to compare all algorithms on the same scale. With this in mind, we used $\gamma = N/M$ for (BL, AB, gAB) and $\gamma = N/D$ for RC. For TG, recall that, when the weights were normalized to sum to one, every $\|\mathbf{q}_i\|^2$ was multiplied by a factor of $1 + 2w_{i\cdot} = 3$ (see Section 3.1.4). In order to compare everything on the same scale, we calibrated this extra factor of 3 back to 1 by choosing $\gamma = N/(3M)$ for TG.

Table 1. Summary of algorithms compared.

Label	Objective function		Other details where applicable			
			γ	c	$w(i,i')$	θ
BL	L_{BL}	Eq. (5)	N/M	—	—	—
AB	L_{AB}	Eq. (12)	N/M	1	—	—
gAB	L_{gAB}	Eq. (14)	N/M	1	Eq. (15)[a]	1
TG	L_{TG}	Eq. (17)	$N/(3M)$	—	Eq. (20)[a]	—
RC	L_{RC}	Eq. (23)	N/D	—	—	—

[a]The weights are normalized to sum to one for every i.

Table 2. Summary statistics for data sets.

	Recipes	Movies
No. of users, N	1706	943
No. of items, M	1040	1682
No. of attributes, D	1057	19
No. of known ratings, $\|T\|$	64 941	100 000
density ratio, $\|T\|/(MN)$	3.7%[a]	6.3%

[a]Notice that this ratio would have been even lower had we used the full recipe data from ref. [18].

4.2. Data sets

We used two data sets—'Recipes' and 'Movies'. The data set, 'Recipes', is a subset of data crawled from http://allrecipes.com/ by Forbes and Zhu [18], including only recipes rated by at least 90 users, and users who rated at least 50 recipes. The data set, 'Movies', is the 'MovieLens 100K' data set from http://www.grouplens.org/.

For 'Recipes', the ratings are integers between 0 and 5, and the binary attribute a_{id} is an indicator of whether recipe i contains ingredient d. For 'Movies', the ratings are integers between 1 and 5, and a_{id} is an indicator of whether movie i belongs to genre d—notice that the same movie can (and often do) belong to multiple genres. Table 2 contains summary statistics about these two data sets.

4.3. Evaluation

To compare and evaluate different algorithms, we repeated the same experiment 15 times. Each time, we sampled 50% of the user-item pairs $(u,i) \in T$ to serve as a hold-out validation set, denoted by T'. Using the remaining 50% of the known ratings, we learned the matrices $\mathbf{P}$ and $\mathbf{Q}$ (or $\mathbf{B}$ in the case of RC) with different algorithms. Ratings for all $(u,i) \in T'$ were predicted by

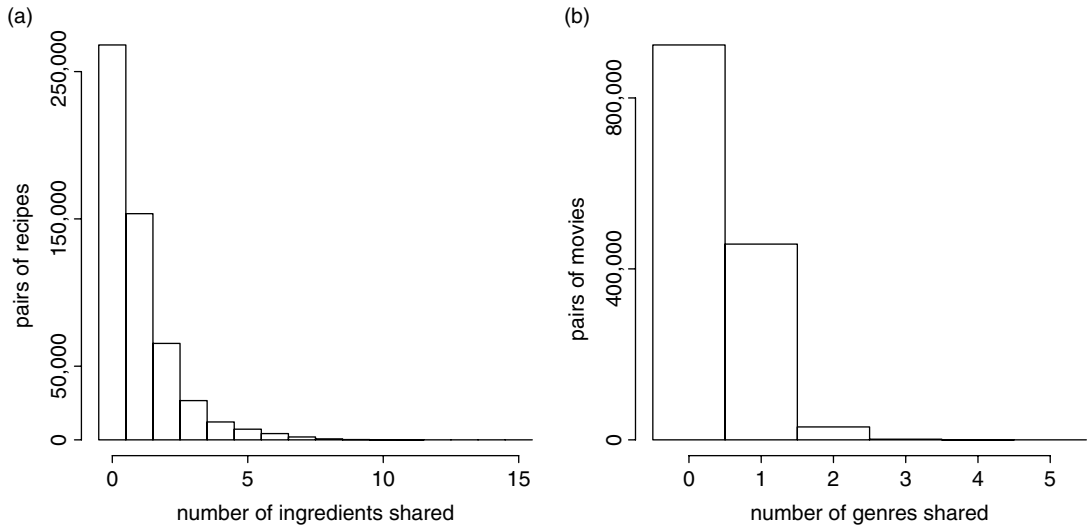

Fig. 2 Distribution of the number of attributes shared by pairs of items. (a) Recipes, (b) Movies.

$\widehat{r}_{ui} = \mathbf{p}_u^{\mathrm{T}}\mathbf{q}_i$ (or $\widehat{r}_{ui} = \mathbf{p}_u^{\mathrm{T}}\mathbf{B}^{\mathrm{T}}\mathbf{a}_i$ in the case of RC)[1]—with proper truncation if $\widehat{r}_{ui}$ fell outside $[0, 5]$ (for 'Recipes') or $[1, 5]$ (for 'Movies')—and evaluated by the mean absolute error (MAE) metric:

$$\mathrm{MAE} = \frac{1}{|T'|}\sum_{(u,i)\in T'} |r_{ui} - \widehat{r}_{ui}|.$$

Many researchers (see refs. [12,29]) have considered the MAE more appropriate for discrete ratings, and the literature is increasingly favoring the use of the MAE as opposed to the RMSE, which dominated the Netflix contest. For each algorithm, we examined factorizations of a few different dimensions, in particular, $K = 5, 10$ and 15.

4.4. Additional Details for AB and gAB

For both data sets, most items do *not* share any attribute in common; for those that do, the number of attributes shared is typically small (see Fig. 2). Thus, we chose $c = 1$ for AB and gAB, activating the alignment penalty as long as two items shared any attribute at all.

Generally speaking, one can certainly regard c as an additional tuning parameter for AB, but if performance is measured with gross overall metrics such as the MAE or the RMSE, then the range of reasonable choices for c is fairly limited in our opinion. We think the best strategy is to choose c so that the alignment penalty is activated for a certain $x\%$ of the item-pairs, and the sensible range for x is somewhere between 10 and 50. If only a handful of item-pairs were subject to the alignment penalty, the overall MAE or RMSE would barely be affected. On the other hand, if more than half of the item-pairs were subject to such a penalty, items would almost certainly be shrunken blindly toward those with which they have little in common. The limited range of sensible values for x and the discrete nature of c often greatly restrict the choice of c. Take the 'Movies' data set, for example. Choosing $c \geq 2$ would have resulted in $x \leq 2.2$, whereas choosing $c = 0$ would have resulted in $x = 100$ (by definition), so the only sensible choice remaining is $c = 1$, which gives $x \approx 35$.

As for gAB, it is clear that a large smoothing parameter θ will cause it to behave very much like AB, whereas a small θ will essentially eliminate the effect of the alignment penalty. To focus on main ideas rather than fine details, we only provide an *illustration* of this algorithm using $\theta = 1$.

4.5. Initialization

We have already mentioned that, when both $\mathbf{P}$ and $\mathbf{Q}$ are unknown, the optimization problem in Eq. (5) is not convex, which means the alternating gradient descent algorithm will give us local solutions at best. Hence, a good initialization strategy is useful.

4.5.1. SVD strategy

For given K, one way to obtain reasonably good initial values of $\mathbf{P}$ and $\mathbf{Q}$ is as follows. First, impute the missing entries of $\mathbf{R}$ with predictions from a certain rudimentary model (more below)—call the resulting matrix

[1] The predicted rating was actually $\widehat{r}_{ui} + \widehat{\mu} + \widehat{\alpha}_u + \widehat{\beta}_i$; see Section 2.2.

$\mathbf{R}_*$. Then, apply regular SVD and obtain the best rank-K approximation to $\mathbf{R}_*$:

$$\mathbf{R}_* \approx \mathbf{P}_*\mathbf{D}_*\mathbf{Q}_*^{\mathrm{T}}.$$

Finally, initialize $\mathbf{P}$ with $\mathbf{P}^{(0)}_{\mathrm{SVD}} = \mathbf{P}_*\mathbf{D}_*^{1/2}$ and $\mathbf{Q}$ with $\mathbf{Q}^{(0)}_{\mathrm{SVD}} = \mathbf{Q}_*\mathbf{D}_*^{1/2}$. In practice, since both $\mathbf{P}^{(0)}$ and $\mathbf{Q}^{(0)}$ are orthogonal matrices, such an initialization strategy is often enough to guard against degeneracy even though the optimization problem in Eq. (5) is somewhat ill-posed without explicit orthogonality constraints (see Section 2.6).

The ANOVA model in Eq. (1) can be used as a rudimentary prediction model for imputing the missing entries. But since the ANOVA model was actually removed prior to the application of any MF techniques (see Section 2.2), all imputed values should just be zero—this would correspond to imputing the missing entries with predictions from the ANOVA model before the normalization took place.

It is easy to see that such an initialization strategy would be applicable to BL, AB, gAB, and TG. For RC, however, an extra step would be required to obtain $\mathbf{B}^{(0)}$ from $\mathbf{Q}^{(0)}$. Since the RC constraint is $\mathbf{Q} = \mathbf{AB}$, the most natural way to do so would be to initialize $\mathbf{B}$ with

$$\mathbf{B}^{(0)}_{\mathrm{SVD}} = \left(\mathbf{A}^{\mathrm{T}}\mathbf{A}\right)^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{Q}^{(0)}_{\mathrm{SVD}}, \tag{28}$$

or, if $D > M$ (in which case $\mathbf{A}^{\mathrm{T}}\mathbf{A}$ would not be invertible),

$$\mathbf{B}^{(0)}_{\mathrm{SVD}} = \left(\mathbf{A}^{\mathrm{T}}\mathbf{A} + \delta\mathbf{I}\right)^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{Q}^{(0)}_{\mathrm{SVD}} \tag{29}$$

for some $\delta > 0$. Our default choice was to set δ to the median value of the diagonal elements in $\mathbf{A}^{\mathrm{T}}\mathbf{A}$.

4.5.2. *Mixed strategy*

While practically useful on its own, the aforementioned SVD strategy posed a subtle problem for comparison: it forced RC into a relative disadvantage. This is because, if $\langle \mathbf{P}^{(0)}_{\mathrm{SVD}}, \mathbf{Q}^{(0)}_{\mathrm{SVD}} \rangle$ is a reasonably good initial factorization of $\mathbf{R}$, then $\langle \mathbf{P}^{(0)}_{\mathrm{SVD}}, \mathbf{AB}^{(0)}_{\mathrm{SVD}} \rangle$ will *not* be as good, since

$$\mathbf{AB}^{(0)}_{\mathrm{SVD}} = \mathbf{A}\left(\mathbf{A}^{\mathrm{T}}\mathbf{A}\right)^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{Q}^{(0)}_{\mathrm{SVD}}$$

is a simply projected version of $\mathbf{Q}^{(0)}_{\mathrm{SVD}}$. Figure 3 provides a geometric explanation of why this is the case.

For a fair comparison of all algorithms, we therefore used a *mixed* strategy for initialization. More specifically, the matrix $\mathbf{P}$ was initialized with

$$\mathbf{P}^{(0)} = \kappa\mathbf{P}^{(0)}_{\mathrm{SVD}} + (1-\kappa)\mathbf{P}^{(0)}_{\mathrm{RANDOM}},$$

where $\mathbf{P}^{(0)}_{\mathrm{SVD}}$ was obtained using the SVD strategy, and $\mathbf{P}^{(0)}_{\mathrm{RANDOM}}$ was a random matrix whose elements were sampled independently from $\mathrm{N}(0, \sigma^2)$. The same procedure was used to initialize $\mathbf{Q}$ and/or $\mathbf{B}$. For given K, the parameters κ and σ were chosen separately for (BL, AB, gAB, TG) and for RC so that the initial factorizations yielded approximately the same level of predictive performance for all the algorithms (see Fig. 4).

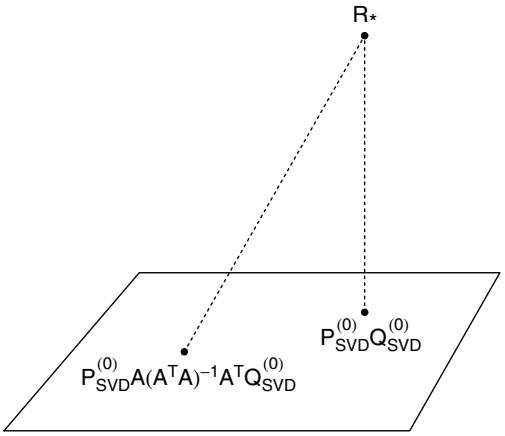

Fig. 3 A geometric explanation of why the SVD initialization strategy forces RC into a relative disadvantage. In this illustration, $\mathbf{P}^{(0)}_{\mathrm{SVD}}$ is fixed; $\mathbf{Q}^{(0)}_{\mathrm{SVD}}$ gives the best factorization of $\mathbf{R}_*$; and anything other than $\mathbf{Q}^{(0)}_{\mathrm{SVD}}$ gives a worse factorization.

4.6. The Choice of λ

Our mixed initialization strategy (see Section 4.5), which ensures that the initial factorization has approximately the same performance for all algorithms, and the way we have scaled the penalty terms (see Section 4.1), so that the penalty on $\sum \|\mathbf{q}_i\|^2$ (or $\sum \|\mathbf{b}_d\|^2$ in the case of RC) is on the same order of magnitude as the penalty on $\sum \|\mathbf{p}_u\|^2$—a quantity that remains constant for all algorithms, imply that, for the purpose of fair comparison, we could (and should) use the same λ for all algorithms.

Table 3 lists the λ's we used for all the experiments. Our λ's increased with K, the dimension (or rank) of the factorization, because more regularization was needed for factorization models that contained more parameters. For any given K, larger λ's were used for the 'Movies' data set than for the 'Recipes' data set because the 'Recipes' data set was more sparse, i.e., the ratio $|T|/N$ was smaller (see Table 2). This meant that, in the case of BL for example, the same level of regularization as measured by the ratio,

$$\frac{\sum_{(u,i)\in T}(r_{ui} - \mathbf{p}_u^{\mathrm{T}}\mathbf{q}_i)^2}{\lambda(\sum_u \|\mathbf{p}_u\|^2 + \gamma \sum_i \|\mathbf{q}_i\|^2)},$$

could be achieved with a smaller λ.

Table 3. The size of the penalty (λ) and the learning rate (η) used for different experiments.

	λ		η ($\times 10^{-3}$)	
K	Movies	Recipes	Movies	Recipes
5	25	8	2.0	2.0
10	50	12	1.0	1.5
15	75	16	0.5	1.0

4.7. Convergence Criterion and the Learning Rate η

All algorithms were presumed to have reached convergence when the percent improvement in their respective objective functions fell below a pre-specified threshold, that is, when

$$\frac{L^{(j)} - L^{(j+1)}}{L^{(j)}} < \varepsilon. \tag{30}$$

We used $\varepsilon = 0.005$ for all algorithms.

For gradient descent algorithms, it is well understood that η should be kept fairly small to ensure that we are moving in a descent direction at each iteration. On the other hand, for practical reasons (e.g., so that the algorithm does not take forever to finish running) we would like to use the largest η feasible—one that still ensures that we are moving downhill. For the convergence criterion in Eq. (30), however, it was critical that the learning rate η did not differ significantly for different algorithms. Suppose Algorithm 1 used a relatively large η and Algorithm 2 used a relatively small one. Then, *relative to* Algorithm 1, Algorithm 2 could 'converge' prematurely according to Eq. (30) simply because the small η did not allow its objective function to change very much from iteration to iteration. Therefore, for any given K, not only did we use the same λ for all algorithms, we also used the same η (see Table 3).

4.8. Results

Figure 4 summarizes our experimental results. We can see that, starting with initial factorizations of roughly the same quality and using the same level of regularization (as controlled by γ and λ), the same learning rate (η), and the same convergence criterion in Eq. (30), the content-boosted algorithms (AB, gAB, TG, RC) generally had lower MAEs than the BL. The performance of TG appears to trail behind that of similar algorithms in the same class (i.e., AB, gAB). We think this is due to its much dampened shrinkage effect (see Section 3.1.4).

5. DISCUSSIONS

We now describe useful by-products from these content-boosted matrix-factorization techniques.

5.1. More Interpretable Recommendations

By explicitly pulling 'similar' items together in the latent feature space, where 'similarity' is defined by the contents of the items, the alignment-biased algorithms (AB, gAB, TG) produce recommendations that are easier to explain. Research has shown that the 'why' dimension of recommendation—the ability 'to reason to the user why certain recommendations are presented' [19]—improves the effectiveness of the recommender system, especially as measured by the conversion rate [20].

To illustrate, we selected a number of movies from a few distinct genres (e.g., thriller, sci-fi), as well as a number of recipes from a few different categories (e.g., soup, pasta, cookie), and plotted their latent feature vectors $\mathbf{q}_i \in \mathbb{R}^5$ from BL and from AB, using the first two principal components (Fig. 5). Here, we chose to illustrate the 5D solutions because showing higher-dimensional solutions in 2D would have created more distortion.

As expected, recipes containing common ingredients—e.g., 'Greek chicken pasta' and 'sesame paste chicken salad'—have been pulled closer together by the alignment-biased algorithm. The two chicken soups are closer to each other. The dish, 'apple stuffed chicken breast', is now closer to chicken pastas than to apple deserts. On the other hand, 'oatmeal raisin cookies' are pulled away from the other two, 'chocolate-chip cookies' because the key ingredients are different.

Similarly, movies belonging to the same genres are now closer to each other, e.g., 'Interview with Vampire' and 'Scream'—both thrillers. The same can be said about the three children's movies and the three science fictions. Clearly, the coordinate maps produced by the alignment-biased algorithm, AB, are much easier to explain to consumers.

5.2. Measure of Content Similarity

The regression-constrained (RC) algorithm allows us to compute the similarity of two content attributes, d and d', using their latent feature vectors, e.g.,

$$\cos(d, d') = \frac{\mathbf{b}_d^{\mathrm{T}} \mathbf{b}_{d'}}{\|\mathbf{b}_d\| \|\mathbf{b}_{d'}\|}, \tag{31}$$

where $\mathbf{b}_d^{\mathrm{T}}$ is the dth row of the matrix $\mathbf{B}$. Notice that, as a measure of similarity, Eq. (31) is *not* based on the simple notion of co-occurrence—merely counting how often two attributes are shared by the same item, since $\mathbf{b}_d$ is driven by both content attributes and user preferences.

Table 4 shows a few examples from both data sets, for pairs of ingredients and genres ranging from being highly similar ($\cos \gg 0$) to being highly dissimilar ($\cos \ll 0$).

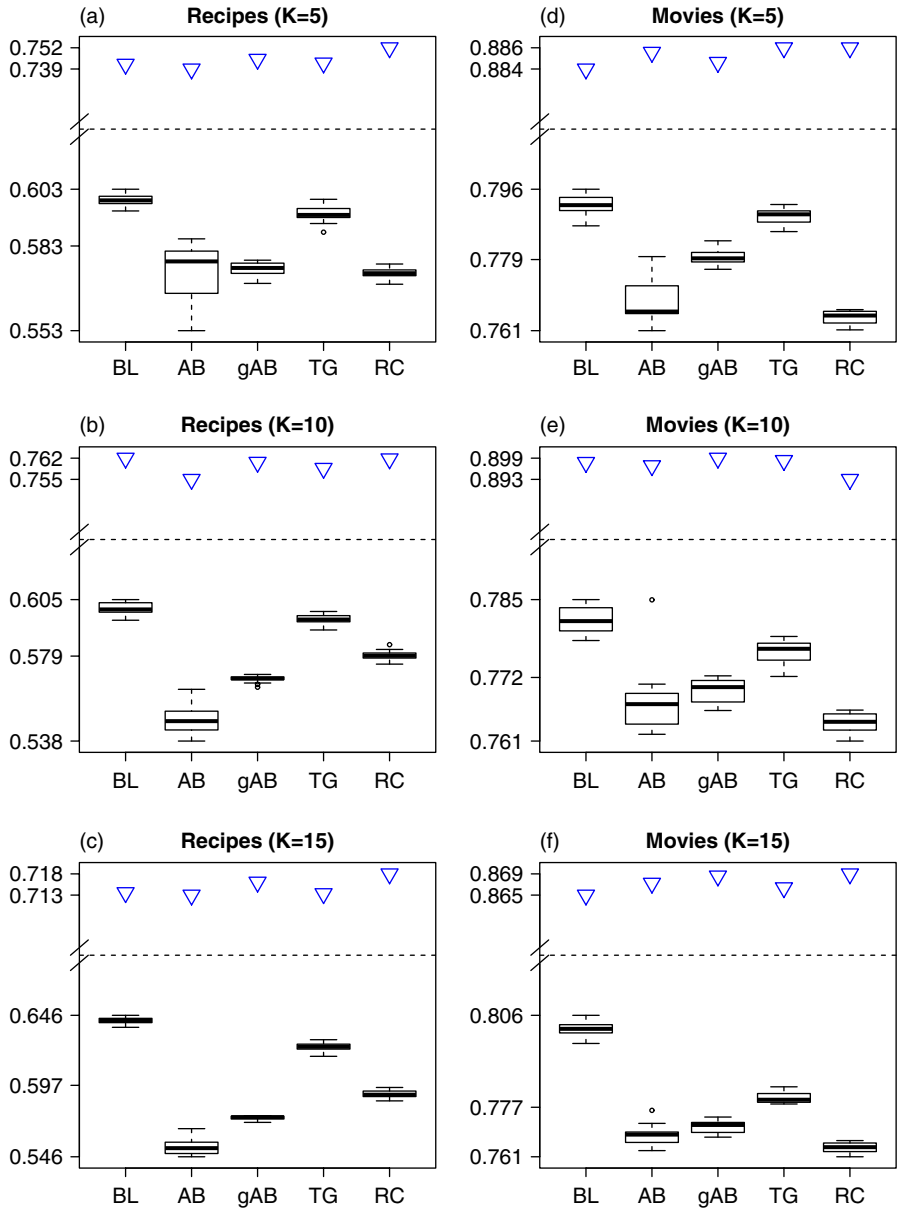

Fig. 4 Mean absolute errors (MAEs) on hold-out validation sets from 15 repeated runs. For each run, the data set was randomly split into a training set and a validation set (see Section 4.3). The inverted triangles ($\triangledown$) on the top indicate the average MAEs on the validation set using the initial values for each respective algorithm, i.e., $\mathbf{P}^{(0)}$ and $\mathbf{Q}^{(0)}$ (or $\mathbf{B}^{(0)}$ in the case of RC). These are shown here to emphasize the fact that all algorithms were started with initial values of approximately the same quality, so that our overall comparison is fair (see Section 4.5). Notice the broken vertical axes. (a) Recipes ($K = 5$), (b) Recipes ($K = 10$), (c) Recipes ($K = 15$), (d) Movies ($K = 5$), (e) Movies ($K = 10$), (f) Movies ($K = 15$). [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

All results in this table are based on $K = 15$. It is well-known that high-dimensional vectors are more likely to be orthogonal ($\cos \approx 0$) than low-dimensional ones. We chose to calculate Eq. (31) using relatively high-dimensional feature vectors so that cosine-values far away from zero were more meaningful.

Some of these pairs are not too surprising. For example, it is easy to see that people who like 'Thai chili sauce' would also like 'jalapeno peppers' (Table 4a)—both spicy ingredients. Similarly, we are hardly amazed that those who like 'crime' movies will probably also like 'horror' movies, and that the genre 'children' goes much better with 'adventure' than with 'documentary' ($\cos \approx 0.74 > 0$ vs. $\cos \approx -0.52 < 0$; Table 4b).

Other pairs, however, are much less obvious. For example, Table 4(b) shows that users who like 'war' movies

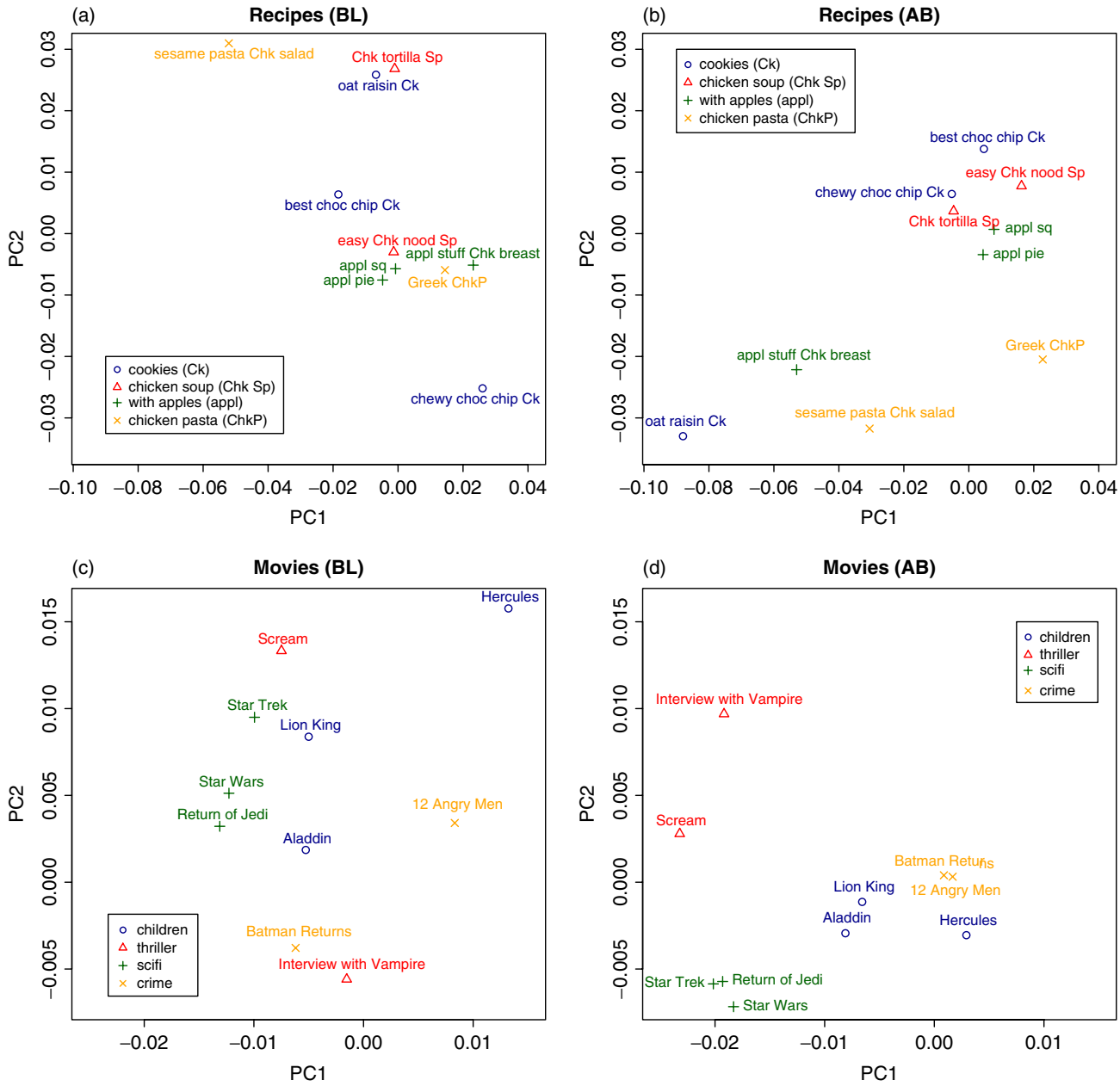

Fig. 5 Feature vectors for selected items—5D matrix factorization solutions projected onto two leading principal components for 2D-display. BL, 'baseline' algorithm (see Section 2); AB, 'alignment-biased' algorithm (see Section 3.1). (a) Recipes (BL), (b) Recipes (AB), (c) Movies (BL), (d) Movies (AB). [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

are more likely to favor 'animation' movies over 'action' movies ($\cos \approx 0.34 > 0$ vs. $\cos \approx -0.21 < 0$). Similarly, Table 4(a) tells us that users who like 'smoked ham' will probably also like 'chocolate mint candy' and that, if a user likes 'cottage cheese', he/she may detest 'Swiss cheese'. This kind of insight about the contents is a unique by-product of the RC algorithm, and some of these novel insights can be commercially useful. For example, Table 4(a) suggests that 'firm tofu' might be used to replace 'mozzarella' in some recipes—if you are familiar with both ingredients, you may very well appreciate that this is not a bad idea at all.

6. FUTURE WORK

Before we end, we would like to briefly mention some open problems for future research.

Table 4. Selected pairs of attributes and their cosine similarity from Eq. (31) based on their latent feature vectors in $\mathbb{R}^{15}$.

(a) Recipes		
Ingredient 1	Ingredient 2	Cosine
Thai chili sauce	Jalapeno peppers	0.9395
Chocolate mint candy	Smoked ham	0.9070
Mozzarella	Firm tofu	0.8828
Cranberry sauce	Ginger garlic paste	0.5283
Almonds	Pork sausage	0.0001
Bread crumbs	Black olive	−0.0001
Cottage cheese	Swiss cheese	−0.5017
Can corn	Golden delicious apple	−0.8709
Condensed milk	Seedless green grapes	−0.8944
Can beef broth	Dry sherry	−0.9402
(b) Movies		
Genre 1	Genre 2	Cosine
Adventure	Children	0.7421
Crime	Horror	0.6456
Action	Sci-fi	0.4354
Animation	War	0.3431
Documentary	Musical	0.0129
Comedy	Film-noir	−0.0160
Action	War	−0.2078
Comedy	Mystery	−0.4348
Children	Documentary	−0.5234
Action	Drama	−0.8973

6.1. Matrix Completion

The CF problem can also be formulated as a *matrix completion* problem [36]:

$$\min_{\widehat{\mathbf{R}}} \quad \text{rank}(\widehat{\mathbf{R}}), \tag{32}$$

$$\text{s.t.} \quad \widehat{r}_{ui} = r_{ui} \quad \text{for} \quad (u,i) \in T. \tag{33}$$

That is, we would like to fill in the missing entries of $\mathbf{R}$ while keeping the rank of the completed matrix $\widehat{\mathbf{R}}$ as low as possible. The rationale behind rank minimization is similar to that behind the matrix factorization approach: we believe that user-preferences are driven by only a few key factors; therefore, the rank of the rating matrix cannot be very high. The two approaches thus share a common philosophical underpinning, but they differ in that the MF approach is more explicit about the nature of the low-rankness.

Suppose $\sigma_1(\widehat{\mathbf{R}}) \geq \sigma_2(\widehat{\mathbf{R}}) \geq \cdots \geq \sigma_{\min(N,M)}(\widehat{\mathbf{R}}) \geq 0$ are the (ordered) singular values of $\widehat{\mathbf{R}}$. Then,

$$\text{rank}(\widehat{\mathbf{R}}) = \sum_{j=1}^{\min(N,M)} I(\sigma_j(\widehat{\mathbf{R}}) \neq 0), \tag{34}$$

and the rank-minimization problem (Eqs. (32 and 33)), like all problems having to do with minimizing the number of non-zero elements (e.g., variable selection [37], compressed sensing [38]), is NP-hard. However, Eq. (34) shows that $\text{rank}(\widehat{\mathbf{R}})$ is equivalent to the l_0-norm of the vector, $\boldsymbol{\sigma} = (\sigma_1, \sigma_2, ..., \sigma_{\min(N,M)})^{\mathrm{T}}$. It has been well established (see refs. [39–41], and references therein) that the l_1-norm is a good convex relaxation of the l_0-norm so, instead of Eq. (32), one can consider

$$\min_{\widehat{\mathbf{R}}} \quad \|\widehat{\mathbf{R}}\|_* \tag{35}$$

where $\|\widehat{\mathbf{R}}\|_*$ denotes the *nuclear-norm* (also known as the *trace-norm*) of $\widehat{\mathbf{R}}$, defined as

$$\|\widehat{\mathbf{R}}\|_* = \sum_{j=1}^{\min(N,M)} \sigma_j(\widehat{\mathbf{R}}),$$

or the l_1-norm of $\boldsymbol{\sigma}$.

The nuclear-norm minimization problem in Eqs. (33)–(35) is convex, and can be solved by semi-definite programming (SDP). Remarkably, under certain conditions, e.g., restricted isometry [42,43] or matrix incoherence [36], the convex optimization problem involving the nuclear-norm (l_1-norm minimization) can be shown to actually solve the minimum-rank problem (l_0-norm minimization). This is an extraordinary set of achievements, made possible by some of the best mathematicians in the world, and it is the main reason why the matrix completion approach is gaining much attention as of late.

Notwithstanding its strong mathematical foundations, the matrix completion approach has some limitations, too. For example, Shi and Yu [44] showed examples where one could fill in a sparse matrix in different ways, while maintaining the same (low) rank for the completed matrix. Computationally, SDP is still only realistic for fairly small matrices, which is why the matrix completion approach has not yet been widely applied to the CF problem, although much research is being devoted to it.

Most importantly for us, however, it is less clear how to generalize the mathematical problem in Eqs. (33)–(35) so as to bring in extra content information—notice that, in our approach, content-based constraints and penalties were introduced to operate directly (and conveniently) on the latent feature vectors, but the matrix completion approach does not admit an explicit parameterization to facilitate the kind of extensions we have proposed. Thus, it appears that, as far as the matrix completion problem is concerned, a different paradigm would be needed for incorporating content information.

6.2. Generative Models

Readers may have noticed the apparent lack of simulation experiments in our study. Simulation experiments are, in

fact, rare in the CF literature. We think this is due to the lack of widely accepted *generative models* for user-rating data. While it is certainly possible to use the latent factor model in Eq. (2) itself to simulate the ratings, this clearly would favor the MF approach in any subsequent performance comparison, and it is easy to understand why the CF community has not found such an approach to be terribly interesting or informative. For our study, the question of what makes a suitable *generative model* is even more complex, since we consider extra content information. Additional research is clearly needed in order to address these issues.

7. SUMMARY

To sum up, we have focused on different ways to incorporate content information directly into the matrix-factorization approach for collaborative filtering. Our methodology consists of imposing either an 'alignment penalty' (see Section 3.1), effectively shrinking items that share common attributes toward each other, or a regression-style constraint (see Section 3.2), forcing the latent item-features to be functions of content attributes. Experiments with two data sets have shown that these content-boosted algorithms can not only achieve better recommendation accuracy, but they can also produce novel, commercially useful insights about the contents themselves, as well as more interpretable recommendations.

Our treatment of the problem is by no means thorough. For example, it is certainly possible to envision different types of penalties and constraints, and we have not yet attempted to study the theoretical properties of these different approaches. As we mentioned in Section 6, it would be interesting to think about how to bring in content information for the matrix completion (as opposed to the MF) approach, and it would be useful to come up with plausible generative models for describing user-rating data. This is a rich area with many opportunities for continued research. We hope that our paper has not only outlined a few useful ideas for practitioners, but also made it easier for researchers to think about this type of problems in a more systematic manner.

ACKNOWLEDGMENTS

This research was partially supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada, and by the University of Waterloo (UW). The first author (J.N.) did most of her work on this project while an undergraduate student at UW. The authors thank Peter Forbes for allowing them to use the recipe data that he assembled, and Laura Ye for creating a denser subset from Peter's original data. The authors also thank three referees and an associate editor for their insightful comments and constructive criticism. They are especially grateful to referee No. 3 for introducing them to the related TG algorithm (see Section 3.1.4).

REFERENCES

[1] A. Feuerverger, Y. He, and S. Khatri, Statistical significance of the Netflix challenge, Stat Sci 27(2) (2012), 202–231.
[2] R. R. Salakhutdinov, A. Mnih, and G. Hinton, Restricted Boltzmann machines for collaborative filtering, In Proceedings of the 24th International Conference on Machine Learning, 2007, 791–798.
[3] Y. Koren, Factorization meets the neighborhood: A multifaceted collaborative filtering model, In Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2008, 426–434.
[4] Y. Koren, R. Bell, and C. Volinsky, Matrix factorization techniques for recommender systems, Computer 42(8) (2009), 30–37.
[5] X. Su and T. M. Khoshgoftaar, A survey of collaborative filtering techniques, Adv Artif Intell 2009. Article ID 421425
[6] S. T. Park, D. Pennock, O. Madani, N. Good, and D. DeCoste, Naïve filterbots for robust cold-start recommendations, In Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2006, 699–705.
[7] Y. Zhao, X. Feng, J. Li, and B. Liu, Shared collaborative filtering, In Proceedings of the 5th ACM Conference on Recommender Systems, 2011, 29–36.
[8] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins, Eigentaste: a constant time collaborative filtering algorithm, Inf Retrieval 4, (2001), 133–151.
[9] A. T. Nguyen, N. Denos, and C. Berrut, Improving new user recommendations with rule-based induction on cold user data, In Proceedings of the 2007 ACM Conference on Recommender Systems, 2007, 121–128.
[10] P. Melville, R. J. Mooney, and R. Nagarajan, Content-boosted collaborative filtering for improved recommendation, In Proceedings of the 18th National Conference on Artificial Intelligence, 2002, 187–192.
[11] G. Semeraro, P. Lops, P. Basile, and M. de Gemmis, Knowledge infusion into content-based recommender systems, In Proceedings of the 3rd ACM Conference on Recommender Systems, 2009, 301–304.
[12] Y. Zhen, W.-J. Li, and D.-Y. Yeung, TagiCoFi: Tag informed collaborative filtering, In Proceedings of the 3rd ACM Conference on Recommender Systems, 2009, 69–76.
[13] M. A. S. N. Nunes, Recommender Systems Based on Personality Traits: Could Human Psychological Aspects Influence the Computer Decision-Making Process?, Berlin, VDM Verlag, 2009.
[14] R. Hu and P. Pu, Enhancing collaborative filtering systems with personality information, In Proceedings of the 5th ACM Conference on Recommender Systems, 2011, 197–204.

[15] M. Jamali and M. Ester, A matrix factorization technique with trust propagation for recommendation in social networks, In Proceedings of the 4th ACM Conference on Recommender Systems, 2010, 135–142.
[16] L. Yu, R. Pan, and Z. Li, Adaptive social similarities for recommender systems, In Proceedings of the 5th ACM Conference on Recommender Systems, 2011, 257–260.
[17] G. Katz, N. Ofek, B. Shapira, L. Rokach, and G. Shani, Using Wikipedia to boost collaborative filtering techniques, In Proceedings of the 5th ACM Conference on Recommender Systems, 2011, 285–288.
[18] P. Forbes and M. Zhu, Content-boosted matrix factorization for recommender systems: experiments with recipe recommendation, In Proceedings of the 5th ACM Conference on Recommender Systems, 2011, 261–264.
[19] N. Sundaresan, Recommender systems at the long tail, In Proceedings of the 5th ACM Conference on Recommender Systems, 2011, 1–5.
[20] R. Sinha and K. Swearingen, The role of transparency in recommender systems, In CHI '02 Extended Abstracts on Human Factors in Computing Systems, 2002, 830–831.
[21] E. Van Buskirk, How the Netflix prize was won, 2009, Wired magazine online, http://www.wired.com/business/2009/09/how-the-netflix-prize-was-won/.
[22] K. V. Mardia, J. T. Kent, and J. M. Bibby, Multivariate Analysis, Academic Press, 1979.
[23] R. R. Salakhutdinov and A. Mnih, Probabilistic matrix factorizations, In Advances in Neural Information Processing Systems, Vol. 20, J. Platt, D. Koller, Y. Singer, and S. Roweis, eds. Cambridge, MA, MIT Press, 2008.
[24] S. Funk, Netflix update: Try this at home (online blog), 2006, http://sifter.org/simon/journal/20061211.html.
[25] C. Volinsky, Email communications, 28 August and 3 September, 2010.
[26] D. D. Lee and H. S. Seung, Learning the parts of objects by non-negative matrix factorization, Nature 401(6755) (1999), 788–791.
[27] J.-P. Brunet, P. Tamayo, T. R. Golub, and J. P. Mesirov, Metagenes and molecular pattern discovery using matrix factorization, Proc Natl Acad Sci USA 101(12) (2004), 4164–4169.
[28] Q. Gu, J. Zhou, and C. Ding, Collaborative filtering: weighted nonnegative matrix factorization incorporating user and item graphs, In Proceedings of the 10th SIAM International Conference on Data Mining, 2010, 199–210.
[29] M. Wu, Collaborative filtering via ensembles of matrix factorizations, In Proceedings of KDD Cup and Workshop, 2007, 43–47.
[30] W.-J. Li and D.-Y. Yeung, Relation regularized matrix factorization, In Proceedings of the 21st International Joint Conference on Artificial Intelligence, 2009, 1126–1131.
[31] J.-P. Benzécri, L'Analyse des Données (Vol. II): L'Analyse des Correspondances, Dunod, Paris, 1973.
[32] M. Greenacre, Theory and Applications of Correspondence Analysis, London, Academic Press, 1983.
[33] C. J. F. ter Braak, Correspondence analysis of incidence and abundance data: properties in terms of a unimodal response model, Biometrics 41 (1985), 859–873.
[34] C. J. F. ter Braak, Canonical correspondence analysis: a new eigenvector technique for multivariate direct gradient analysis, Ecology 67(5) (1986), 1167–1179.
[35] C. J. F. ter Braak, Unimodal Models to Relate Species to Environment, Wageningen, DLO-Agricultural Mathematics Group, 1996.
[36] E. J. Candès and B. Recht, Exact matrix completion via convex optimization, Found Comput Math 9 (2009), 717–772.
[37] R. Tibshirani, Regression shrinkage and selection via the Lasso, J Roy Stat Soc Ser B, 58 (1996), 267–288.
[38] D. L. Donoho, Compressed sensing, IEEE Trans Inf Theory 52(4) (2006), 1289–1306.
[39] D. L. Donoho, For most large underdetermined systems of linear equations the minimal l_1-norm solution is also the sparsest solution, Commun Pure Appl Math 59(6) (2006), 797–829.
[40] E. J. Candès and T. Tao, Near-optimal signal recovery from random projections: universal encoding strategies?, IEEE Trans Inf Theory 52(12) (2006), 5406–5425.
[41] E. J. Candès, J. K. Romberg, and T. Tao, Stable signal recovery from incomplete and inaccurate measurements, Commun Pure Appl Math 59(8) (2006), 1207–1223.
[42] E. J. Candès and T. Tao, Decoding by linear programming, IEEE Trans Inf Theory 51(12) (2005), 4203–4215.
[43] B. Recht, M. Fazel, and P. A. Parrilo, Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization, SIAM Rev 52 (2010), 471–501.
[44] X. Shi and P. S. Yu, Limitations of matrix completion via trace norm minimization, SIGKDD Explorations 12(2) (2010), 16–20.