Content-boosted Matrix Factorization for Recommender Systems: Experiments with Recipe Recommendation*

Peter Forbes†
Department of Pure Mathematics and Mathematical Statistics, University of Cambridge, United Kingdom
p.forbes@statslab.cam.ac.uk

Mu Zhu
Department of Statistics and Actuarial Science, University of Waterloo, Canada
m3zhu@math.uwaterloo.ca

ABSTRACT

The Netflix prize has rejuvenated a widespread interest in the matrix factorization approach for collaborative filtering. We describe a simple algorithm for incorporating content information *directly* into this approach. We present experimental evidence using recipe data to show that this not only improves recommendation accuracy but also provides useful insights about the contents themselves that are otherwise unavailable.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data Mining*; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Information Filtering*; I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms, Experimentation, Performance

1. INTRODUCTION

Mainstream recommender systems today — such as those used by Netflix (`www.netflix.com`), Amazon (`www.amazon.com`), and Pandora (`www.pandora.com`) — often take one of the following two approaches:

1. *Collaborative filtering approach* (Netflix). Based on the movies you liked and disliked, we have found users of similar tastes. Since they liked the following movies, we think you may like them too, even though we have no idea what types of movies they are.
2. *Content-based approach* (Pandora). Based on the songs you liked and disliked, it appears that you like slow, soft songs sung by a low, female voice. By analyzing all the songs in our digital library, these appear to be slow, soft songs sung by a low, female voice, and we think you will like them.

The collaborative filtering (CF) approach has the advantage of not requiring machine analyzable content; thus it is capable of recommending an item without understanding the item itself [5]. But, for the very same reason, it suffers from the so-called "cold start" problem — predictions for newer items that have not received much user feedback tend to be very inaccurate. However, this problem can be mitigated to some extent by enhancing CF to exploit any known content information. Melville *et al.* [4] developed such a hybrid, content-boosted CF system by taking a two-step approach. They first filled in the sparse user rating matrix $\mathbf{S}$ (see § 2 below) with predictions from a purely content-based classifier, and then applied a CF algorithm to the resulting dense matrix.

In this paper, we describe and experiment with a simple algorithm for incorporating content information *directly* into the matrix factorization approach [3], which became popular due to the recent Netflix contest (`www.netflixprize.com`). Whereas Melville *et al.* [4] included content information as an intermediate step, we instead incorporate such information *directly* as a natural linear constraint to the matrix factorization algorithm. We present evidence to show that our content-boosted algorithm can provide informative insights about the contents themselves that would otherwise be unavailable.

2. PRELIMINARIES

Most publically available data sets have no content information associated with the items. Thus, we obtained our data by crawling the website, `allrecipes.com`, ourselves. Recipe data are naturally suited for our purposes because, by definition, we know the ingredients used in each recipe, and these ingredient lists provide highly descriptive content information.

Our data contained about $565,000$ scores submitted by about $61,000$ users for about $46,000$ recipes as of May 2010. Overall, these $46,000$ recipes contained about $16,000$ unique ingredients, with each recipe containing 7 ingredients on average. We considered only the top $7,000$ ingredients that were used in ≥ 3 recipes.

Recipe recommendation has been the subject of some previous work. For example, van Pinxteren *et al.* [7] developed

*Funding provided by NSERC and University of Waterloo.
†Corresponding author.

RecSys'11, October 23–27, 2011, Chicago, Illinois, USA.

an algorithm to classify 8,701 recipes using 55 predetermined features, whereas Freyne and Berkovsky [1] used the ingredients themselves as features for 183 recipes and recommendations were selected to maximize the number of highly-scored ingredients. While both projects used the weighted average to determine feature preferences and make recommendations, we study the more sophisticated matrix factorization approach and work with a much larger data set.

Though we will focus on the recipe data for the rest of this paper, we'd like to emphasize that our methodology and ideas are general, whenever content information similar to the ingredient list is available. Here is a summary list of some of our key notations:

- n_u – number of users;
- n_r – number of recipes;
- n_i – number of ingredients;
- $\mathbf{S}$ – an $n_u \times n_r$ matrix, where each entry s_{ur}, if not missing, is the score (in our case, an integer between 0 and 5) given to recipe r by user u;
- $\mathbb{L}$ – index of learning set, i.e., set of (u, r) such that s_{ur} is observed (see §4 and Fig. 1);
- $\mathbb{L}_r$ – set of u such that s_{ur} is observed, for a given r;
- $\mathbb{L}_u$ – set of r such that s_{ur} is observed, for a given u;
- $\mathbb{T}$ – index of test set, i.e., set of (u, r) such that s_{ur} is observed but *pretended* to be "missing" by the collaborative filtering algorithms (see §4 and Fig. 1);
- $\mathbf{X}$ – an $n_r \times n_i$ matrix, where each entry is defined as

$$x_{ri} = \begin{cases} 1, & \text{if recipe } r \text{ contains ingredient } i; \\ 0, & \text{otherwise.} \end{cases}$$

Typical collaborative filtering algorithms work with the matrix $\mathbf{S}$ alone. The extra content information for the items — in our case, recipes — is stored in the matrix $\mathbf{X}$.

All our algorithms used a standardized version of the matrix $\mathbf{S}$, created by removing both column means and row means. These are also called "item biases" and "user biases" in the literature [see, e.g., 3]; they incorporate the obvious knowledge that some recipes are simply better liked than others, and that some users are simply easier to please.

3. ALGORITHM

Experiences from the Netflix contest established the superiority of the matrix factorization approach [3] for collaborative filtering when dealing with large, real-world, noisy data sets.

For a given dimension, n_f, the matrix factorization method aims to factor $\mathbf{S}$ into

$$\mathbf{S} \approx \mathbf{U}\mathbf{R}^T = \underbrace{\begin{bmatrix} \boldsymbol{\mu}_1^T \\ \boldsymbol{\mu}_2^T \\ \vdots \end{bmatrix}}_{n_u \times n_f} \underbrace{\begin{bmatrix} \boldsymbol{\rho}_1 & \boldsymbol{\rho}_2 & \cdots \end{bmatrix}}_{n_f \times n_r}, \tag{1}$$

where $\mathbf{U}$ is an $n_u \times n_f$ matrix whose u-th row is a feature vector $\boldsymbol{\mu}_u \in \mathbb{R}^{n_f}$ for user u, and $\mathbf{R}$ is an $n_r \times n_f$ matrix whose r-th row is a feature vector $\boldsymbol{\rho}_r \in \mathbb{R}^{n_f}$ for recipe r. The objective is to find feature vectors, $\boldsymbol{\mu}_u$ for each user u and $\boldsymbol{\rho}_r$ for each recipe r, such that $\hat{s}_{ur} = \boldsymbol{\mu}_u^T \boldsymbol{\rho}_r$ estimates s_{ur}. In practice, the score s_{ur} often takes on integer values $c, c+1, ..., d$, so $\hat{s}_{ur}$ is truncated to lie within $[c, d]$ before being compared to s_{ur}.

In principle, this factorization can be achieved by considering the optimization problem,

$$\min_{\mathbf{U},\mathbf{R}} \quad \|\mathbf{S} - \mathbf{U}\mathbf{R}^T\|^2,$$

where $\|\cdot\|$ denotes the Frobenius norm. In practice, it is common to put regularization penalties on $\mathbf{U}$ and $\mathbf{R}$ in order to avoid overfitting, for example,

$$\min_{\mathbf{U},\mathbf{R}} \quad \|\mathbf{S} - \mathbf{U}\mathbf{R}^T\|^2 + \lambda \left(\|\mathbf{U}\|^2 + \|\mathbf{R}\|^2 \right). \tag{2}$$

However, since most entries of $\mathbf{S}$ are unknown, we can only compute the first Frobenius norm partially by summing over all known entries of s_{ur}. This changes (2) into

$$\min_{\boldsymbol{\mu}_u, \boldsymbol{\rho}_r} \quad \sum_{(u,r) \in \mathbb{L}} \left(s_{ur} - \boldsymbol{\mu}_u^T \boldsymbol{\rho}_r \right)^2 + \lambda \left(\sum_{u=1}^{n_u} \|\boldsymbol{\mu}_u\|^2 + \sum_{r=1}^{n_r} \|\boldsymbol{\rho}_r\|^2 \right), \tag{3}$$

which is typically solved by an alternating gradient descent algorithm, moving along the gradient with respect to $\boldsymbol{\mu}_u$ while fixing $\boldsymbol{\rho}_r$ and vice versa. That is, we iterate

$$\boldsymbol{\mu}_u \leftarrow \boldsymbol{\mu}_u + \eta \left(\sum_{r \in \mathbb{L}_u} (s_{ur} - \boldsymbol{\mu}_u^T \boldsymbol{\rho}_r) \boldsymbol{\rho}_r - \lambda \boldsymbol{\mu}_u \right) \tag{4}$$

$$\boldsymbol{\rho}_r \leftarrow \boldsymbol{\rho}_r + \eta \left(\sum_{u \in \mathbb{L}_r} (s_{ur} - \boldsymbol{\mu}_u^T \boldsymbol{\rho}_r) \boldsymbol{\mu}_u - \lambda \boldsymbol{\rho}_r \right) \tag{5}$$

until convergence, where η is the step size or learning rate, which we set sufficiently small to ensure convergence.

To exploit the extra content information $\mathbf{X}$ for the matrix factorization method, we constrained the feature vector of each recipe to depend explicitly on its ingredients, i.e.,

$$\mathbf{R} = \mathbf{X}\boldsymbol{\Phi}, \tag{6}$$

where $\boldsymbol{\Phi}$ is an $n_i \times n_f$ matrix whose i-th row is a feature vector $\boldsymbol{\phi}_i \in \mathbb{R}^{n_f}$ for ingredient i. A constraint of this kind was also considered by environmental ecologists to extend correspondence analysis to canonical correspondence analysis by requiring the otherwise latent environmental gradient to explicitly depend on environmental measurements [6].

While this was certainly not the only way to incorporate content information into the matrix factorization approach, we will see later (§5) that it led to interesting and useful insights about the contents themselves in practice.

Under the constraint (6), model (1) became

$$\mathbf{S} \approx \mathbf{U}\boldsymbol{\Phi}^T\mathbf{X}^T = \underbrace{\begin{bmatrix} \boldsymbol{\mu}_1^T \\ \boldsymbol{\mu}_2^T \\ \vdots \end{bmatrix}}_{n_u \times n_f} \underbrace{\boldsymbol{\Phi}^T}_{n_f \times n_i} \underbrace{\begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots \end{bmatrix}}_{n_i \times n_r},$$

which changed (3) into

$$\min_{\boldsymbol{\mu}_u, \boldsymbol{\Phi}} \quad \sum_{(u,r) \in \mathbb{L}} \left(s_{ur} - \boldsymbol{\mu}_u^T \boldsymbol{\Phi}^T \mathbf{x}_r \right)^2 + \lambda \left(\sum_{u=1}^{n_u} \|\boldsymbol{\mu}_u\|^2 + \|\boldsymbol{\Phi}\|^2 \right). \tag{7}$$

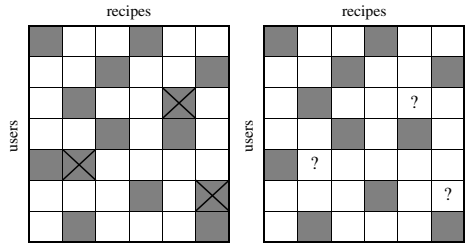

Figure 1: *Experimental set-up.* Left*: Some entries of* $\mathbf{S}$ *are observed (grey), while others are missing (white). Of the observed ones,* 50% *are randomly selected as test data (crosses) and treated as if "missing" by the CF algorithm.* Right*: The CF algorithm learns from all the observed (grey) entries and makes a prediction for every missing (white) entry, but we can only evaluate its performance (e.g., calculate the RMSE) using entries* pretended *to be "missing" (question mark).*

Using the fact that $d(\mathbf{p}^T\mathbf{Y}\mathbf{q})/d\mathbf{Y} = \mathbf{p}\mathbf{q}^T$, we obtained the following alternating gradient-descent equations:

$$\boldsymbol{\mu}_u \leftarrow \boldsymbol{\mu}_u + \eta\left(\sum_{r\in\mathbb{L}_u}(s_{ur} - \boldsymbol{\mu}_u^T\boldsymbol{\Phi}^T\mathbf{x}_r)\boldsymbol{\Phi}^T\mathbf{x}_r - \lambda\boldsymbol{\mu}_u\right) \quad (8)$$

$$\boldsymbol{\Phi} \leftarrow \boldsymbol{\Phi} + \eta\left(\sum_{(u,r)\in\mathbb{L}}(s_{ur} - \boldsymbol{\mu}_u^T\boldsymbol{\Phi}^T\mathbf{x}_r)\mathbf{x}_r\boldsymbol{\mu}_u^T - \lambda\boldsymbol{\Phi}\right) \quad (9)$$

Notice that, unlike the vector equations (4)-(5) and (8), (9) is a matrix equation.

4. EXPERIMENTS

All experiments were repeated 25 times, each time using a random subset drawn without replacement from the whole data set. That is, the matrix $\mathbf{S}$ used to conduct each experiment was somewhat smaller than the whole data set. For each experiment, 50% of the observed entries in $\mathbf{S}$ were randomly selected as test data, indexed by the set $\mathbb{T}$ (see §2 and Fig. 1). Tuning parameters — such as λ in (3) and (7) — were chosen by cross validation on the learning data alone.

We experimented with a range of values for n_f, the dimension of the latent features. The algorithms' performances were then evaluated as a function of n_f; see Fig. 2(A).

To evaluate the algorithms' performances, we followed Koren *et al.* [3] and used the root mean-squared error (RMSE) on $\mathbb{T}$, i.e.,

$$\text{RMSE} = \sqrt{\frac{1}{|\mathbb{T}|}\sum_{(u,r)\in\mathbb{T}}(s_{ur} - \hat{s}_{ur})^2},$$

where $\hat{s}_{ur}$ denotes the score predicted by the algorithm under consideration, and $|\mathbb{T}|$ is the size of the test set $\mathbb{T}$.

Fig. 2(A) shows our experimental results. The vertical axis is the difference in RMSE between the content-boosted algorithm and the original algorithm, so a negative value means the content-boosted algorithm produced a better result. From Fig. 2(A), we see that it is beneficial for the matrix factorization approach to take content information into account. Standard deviations calculated from our 25 repeated experiments also indicate that the improvement was statistically significant.

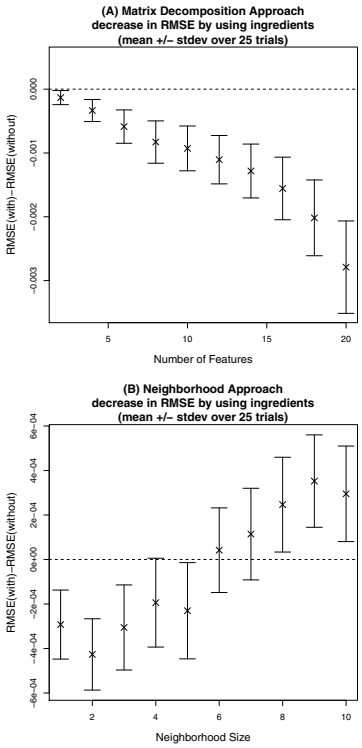

Figure 2: *Experimental results. Differences in RMSE on the test set between an algorithm that uses the content information (denoted by the word, "with") and one that does not (denoted by the word, "without"). A negative value means the content-boosted algorithm did better.*

5. INFERRED CONTENT SIMILARITY

A nice by-product of taking the ingredients into account in the matrix factorization approach is that we can compute the similarity of two ingredients, i and i', using their latent feature vectors, e.g.,

$$\cos(i,i') = \frac{\boldsymbol{\phi}_i^T\boldsymbol{\phi}_{i'}}{\|\boldsymbol{\phi}_i\|\|\boldsymbol{\phi}_{i'}\|}. \quad (10)$$

One obvious way to measure ingredient similarity is to simply count how often two ingredients appear together in the same recipe. In comparison, (10) is based on which ingredients tend to be preferred by the same users, which is very different from the common notion of co-occurrence. We present evidence below that this more sophisticated measure allows us to unearth valuable trends in user tastes.

The top ten pairs of ingredients that appeared most similar in the 20-dimensional latent feature space are shown in Table 1. Many of these pairs seem intuitive (e.g., chocolate cake mix and chocolate frosting). This suggests that the feature vectors $\boldsymbol{\phi}_i$ have successfully captured quantitative features describing different taste preferences. On the other hand, some pairs are not as obvious, such as vodka and watermelon. Results such as this may have commercial appeal. For instance, vodka companies may consider creating a watermelon flavored drink, and grocery stores may gain an advantage by displaying them nearby each other, such as the now famous "beer and diapers" legend.

To compare, we also computed ingredient similarity based on the relative frequency of them appearing together in the

Table 1: *Top-ten ingredient pairs based on their similarity as measured by (10) in a 20-dimensional latent feature space.*

Ingredient 1	Ingredient 2	Cosine
corned beef brisket	Irish stout beer	0.9998
pita bread	lettuce	0.9997
dry red wine	filet mignon steak	0.9994
chocolate cake mix	chocolate frosting	0.9993
chocolate candy bar	whipped topping	0.9990
cinnamon red hot candy	marshmallows	0.9978
vodka	watermelon	0.9977
baking chocolate	Oreo cookies	0.9975
avocado	taco seasoning	0.9972
salmon	onion	0.9967

same recipe. As many would suspect, the top-10 pairs were all combinations of flour, salt, eggs, butter and sugar — not quite as interesting or informative as the ones in Table 1. The top-25 pairs included pepper, vanilla, milk, water, and baking powder, while various combinations of garlic, onion, celery, oil, and shortening rounded out the top 100.

To further confirm that (10) is a better measure of ingredient similarity than counting how often two ingredients would co-appear in the same recipe, we experimented with incorporating content information into another widely-used and perhaps more intuitive approach for collaborative filtering, the nearest neighbors method [2].

For any pair of recipes, say r and r', let

$$w_{r,r'} = \frac{\sum(s_{ur} - \bar{s}_{\cdot r})(s_{ur'} - \bar{s}_{\cdot r'})}{\sqrt{\sum(s_{ur} - \bar{s}_{\cdot r})^2 \sum(s_{ur'} - \bar{s}_{\cdot r'})^2}} \tag{11}$$

be their correlation. Clearly, this can realistically be calculated only with users who have scored both r and r'. In other words, all the summations above are taken over $u \in \mathbb{L}_r \cap \mathbb{L}_{r'}$; and $\bar{s}_{\cdot r}$ is the mean of s_{ur} over the same set, $u \in \mathbb{L}_r \cap \mathbb{L}_{r'}$.

For any given user u, let

$$w_u(r, K) \equiv \text{the } K\text{-th largest value of } w_{r,r'} \ \forall \ r' \in \mathbb{L}_u.$$

To predict whether a user will like a certain recipe, the nearest neighbors method simply computes the weighted average of a few neighboring recipes already scored by this user,

$$\hat{s}_{ur} = \frac{\sum_{r' \in N_u(r)} w_{r,r'} s_{ur'}}{\sum_{r' \in N_u(r)} w_{r,r'}}, \tag{12}$$

where

$$N_u(r) = \{r' \in \mathbb{L}_u \quad \text{such that} \quad w_{r,r'} \geq w_u(r, K)\}.$$

To exploit the extra content information $\mathbf{X}$ for the nearest neighbors method, we replaced equation (11) with

$$w_{r,r'} = \frac{\mathbf{x}_r^T \mathbf{x}_{r'}}{\|\mathbf{x}_r\| \|\mathbf{x}_{r'}\|}, \tag{13}$$

which, in our case (see §2), is easily seen to be equal to

$$w_{r,r'} = \frac{(\#\text{ ingredients in both } r \text{ and } r')}{\sqrt{(\#\text{ ingredients in } r)}\sqrt{(\#\text{ ingredients in } r')}}.$$

That is, neighbors were determined by how many ingredients they shared in common.

We experimented with a range of values for K, the number of neighbors, and the algorithms' performances were evaluated as a function of K; see Fig. 2(B). Here, we see that the benefit of taking content information into account for the nearest neighbors approach was not nearly as substantial as for the matrix factorization approach. The best improvement on the RMSE scale was smaller by a factor of 5, and the content-based neighborhood algorithm actually performed worse for large K. This is because defining recipe neighborhood by the number of ingredients they share in common is not all that helpful for recommendation. For example, a pecan cake and an apple pie necessarily share many ingredients in common — uninteresting ones such as flour, sugar, and butter, but the one main ingredient (pecan versus apple) makes them very different recipes.

Consequently, measuring ingredient similarity by how often they appear together in the same recipe is not all that meaningful, either. The similarity measure (10) — available only as a by-product of our content-boosted matrix factorization algorithm — is much more useful, e.g., it could provide valuable information for us to consider swapping ingredients and creating recipe variations for people with certain food allergies or other dietary constraints.

6. SUMMARY

In this paper, we described a simple, content-boosted matrix factorization algorithm for collaborative filtering (§3). Our algorithm incorporates content information *directly* as a linear constraint. We conducted some experiments using recipe data, and our results confirmed the potential usefulness of our method (§4). We also discussed how our content-boosted algorithm could give us additional insights about the contents themselves that would otherwise not be available, as well as conducted further analyses and experiments to show that these insights were far more informative than those we could obtain by analyzing the contents alone (§5).

References

[1] Freyne, J. and Berkovsky, S. (2010). Intelligent food planning: Personalized recipe recommendation. In *Proceedings of the 15th International Conference on Intelligent User Interfaces*, pages 321–324.

[2] Koren, Y. (2008). Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 426–434.

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

[4] Melville, P., Mooney, R. J., and Nagarajan, R. (2002). Content-boosted collaborative filtering for improved recommendation. In *Proceedings of the 18th National Conference on Artificial Intelligence*, pages 187–192.

[5] Su, X. and Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, **2009**. Article ID 421425.

[6] ter Braak, C. J. F. (1986). Canonical correspondence analysis: A new eigenvector technique for multivariate direct gradient analysis. *Ecology*, **67**(5), 1167–1179.

[7] van Pinxteren, Y., Geleijnse, G., and Kamsteeg, P. (2011). Deriving a recipe similarity measure for recommending healthful meals. In *Proceedings of the 16th International Conference on Intelligent User Interfaces*, pages 105–114.