

New Recommendation Techniques for Multicriteria Rating Systems

Gediminas Adomavicius and YoungOk Kwon, *University of Minnesota*

Personalization technologies and recommender systems help online consumers avoid information overload by making suggestions regarding which information is most relevant to them. Most online shopping sites and many other applications now use recommender systems. The most popular examples include Netflix, which recommends movies, and Amazon.

com, which recommends books, CDs, and various other products. Users offer feedback on purchased or consumed items, and the recommender system uses the information to predict their preferences for yet unseen items and subsequently recommends items with the highest predicted relevance.

The vast majority of current recommender systems use a single criterion, such as a single numerical rating, to represent an item's utility to a user in a $2D\ Users \times Items$ space. Single-criterion rating systems have proved successful in several applications, but many industries have begun employing multicriteria systems. For example, restaurant guides, such as Zagat's Guide, provide three criteria for restaurant ratings (food, décor, and service). Online shopping malls, such as Circuitcity.com and Buy.com, use multicriteria ratings for consumer electronics (display, performance, battery life, and cost). However, these rating systems are not used in the context of personalization; the rating on each criterion—for example, Zagat's "food" rating for a specific restaurant—is the same for all users. In contrast, Yahoo! Movies recently launched a recommendation service that employs user-specific multicriteria ratings for each movie. This move indicates that multicriteria data provides value to online content providers and consumers as a component in personalization applications.

Taking full advantage of multicriteria ratings in personalization applications requires new recommendation techniques. In this article, we propose several new techniques for extending recommendation technologies to incorporate and leverage multicriteria rating information (see the sidebar for related work).

General characteristics of recommender systems

Recommender systems are usually classified according to their recommendation approach:¹

- *Content-based* approaches recommend items similar to those the user preferred in the past.
- *Collaborative-filtering* approaches recommend items that users with similar preferences have liked in the past.
- *Hybrid* approaches combine content-based and collaborative-filtering methods in several different ways.²

We can also classify recommender systems according to their algorithmic technique:³

- *Memory-based* techniques usually represent heuristics that calculate recommendations on the fly based directly on previous user activities.
- *Model-based* techniques use previous user activities to first learn a predictive model (typically using some statistical or machine-learning method), which the system then uses to make recommendations.

A comprehensive survey of recommender systems research literature is beyond the scope of this article, although a recent survey is available elsewhere.² Typically, the recommendation process starts with the specification of an initial ratings set, which either the user explicitly provides or the system implicitly infers. For example, in a movie recommender sys-

Two new recommendation techniques leverage multicriteria ratings and improve recommendation accuracy as compared with single-rating recommendation approaches.

tem, John Doe might assign a single-criterion rating of 11 (out of 13) for *Vertigo*; in other words, $R(\text{John_Doe}, \text{Vertigo}) = 11$. Once the initial ratings are specified, a recommender system tries to estimate the rating function R

$$R: \text{Users} \times \text{Items} \rightarrow R_0 \quad (1)$$

for the (user, item) pairs that the user has not yet rated. R_0 is a set of values that a predicted rating can take; it is usually represented by a totally ordered set—for example, integers or real numbers within a certain range.

Once the system estimates function R , it can recommend the highest-rated item (or a set of N highest-rated items) for each user. So, one goal of a typical recommender system is to correctly estimate the ratings of unrated items on the basis of given ratings; another goal is to find items that maximize the utility for each user.

Traditional collaborative-filtering approach

Before proceeding with the discussion of new multicriteria rating techniques, we briefly describe one of the traditional and commonly used single-rating collaborative recommendation techniques. We will use this technique as an example throughout this article.

Specifically, using the recommender systems classification schemes we have described, let's consider a memory-based collaborative-filtering technique that estimates $R(u, i)$ —the rating that user u would give to item i —by computing the weighted average of all known ratings $R(u', i)$, where user u' is “similar” to u . We consider two popular ways to compute this weighted average.³ The first is the *weighted sum approach*:

$$R(u, i) = z \sum_{u' \in N(u)} \text{sim}(u, u') \cdot R(u', i) \quad (2)$$

and the second is the *adjusted weighted sum approach*:

$$R(u, i) = \overline{R(i)} + z \sum_{u' \in N(u)} \text{sim}(u, u') \cdot (R(u', i) - \overline{R(u')}) \quad (3)$$

Both methods weight the value of $R(u', i)$ by the similarity of user u' to user u —the more similar the two users are, the more weight $R(u', i)$ will have in computing the value of $R(u, i)$. Furthermore, multiplier z serves as a normalizing factor and is usually set to

$$z = \frac{1}{\sum_{u' \in N(u)} |\text{sim}(u, u')|}$$

$\overline{R(u)}$ represents the average rating of user u , and $N(u)$ represents the set of users that are similar to user u . Set $N(u)$ can range in size anywhere from 1 to all users in the data set. Limiting the neighborhood size to some specific number—say, 3—will determine how many similar users the computation of rating $R(u, i)$ will use.

Furthermore, there are several ways to compute similarity $\text{sim}(u, u')$ between two users, including cosine-based and correlation-based computations.³ We'll use the *cosine-based similarity* here, because it is arguably the most commonly used technique for determining how similar two users are in memory-based collaborative-filtering algorithms. Assuming $I(u, u')$ represents the set of all items rated by both users u and u' , we can calculate the cosine-based similarity as follows:

$$\text{sim}(u, u') = \frac{\left(\sum_{i \in I(u, u')} R(u, i) R(u', i) \right)}{\left(\sqrt{\sum_{i \in I(u, u')} R(u, i)^2} \sqrt{\sum_{i \in I(u, u')} R(u', i)^2} \right)} \quad (4)$$

In addition, because of the inherent symmetry between users and items in the traditional memory-based collaborative-filtering setting, this approach can be either *user-based* or *item-based*, depending on whether we want to calculate the similarity between users or items. Equations 2 and 3 represent the user-based approach, but rewriting them for the item-based approach is straightforward. For example, we can calculate the item-based adjusted weighted sum as follows:⁴

$$R(u, i) = \overline{R(i)} + z \sum_{i' \in N(i)} \text{sim}(i, i') \cdot (R(u, i') - \overline{R(i')}) \quad (5)$$

and z , $\overline{R(i)}$, $\text{sim}(i, i')$, and $N(i)$ are analogous to their user-based counterparts.

In the rest of the article, unless explicitly stated otherwise, “traditional collaborative filtering approach” refers to the *user-based adjusted weighted sum* approach (equation 3) that uses the cosine-based similarity function (equation 4).

Incorporating multicriteria ratings into recommender systems

In addition to the overall rating, multicriteria ratings provide information about user preferences for different aspects or components of an item. Recommender systems should benefit from leveraging this additional information because it can potentially increase the recommendation accuracy. Therefore, we need new techniques to effectively incorporate the multicriteria rating information into the recommendation process.

Multicriteria recommender systems aim to find items that are most useful to each user, just as single-rating recommender systems do. So, the system must be able to predict each item's overall rating for each user so that it can compare the items on the basis of their overall ratings and recommend the best items to the users. The difference between single-rating and multicriteria rating systems is that the latter have more information about the users and items to use in the recommendation process. More formally, the general form of a rating function in a multicriteria recommender system is

$$R: \text{Users} \times \text{Items} \rightarrow R_0 \times R_1 \times \dots \times R_k \quad (6)$$

where R_0 is the set of possible overall rating values, and R_i represents the possible rating values for each individual criterion i ($i = 1, \dots, k$), typically on some numeric scale (for example, from 1 to 13).

We propose two new recommendation approaches and present several different variations of each. The first approach extends the traditional single-criteria memory-based collaborative-filtering algorithm, while the second approach has no restriction to any specific algorithm. In other words, it can use any existing single-criteria recommendation algorithm—content-based, collaborative, or hybrid.

Similarity-based approach

Consider a movie recommendation application, where users provide

Related Work in Multicriteria Decision Problems

Research in multicriteria problems is extensive in both operations research and decision science fields. Most engineering problems are essentially multicriteria optimization problems.¹ For example, when engineers design an airplane, they consider its reliability, longevity, efficiency, cost, and other utilization factors. Typical methods to solve the multicriteria optimization problems include finding Pareto optimal solutions; optimizing the most important criterion and converting other criteria to constraints; and consecutively optimizing one criterion at a time, then converting an optimal solution to constraints and repeating the process for other criteria.

The decision science field treats organizational decision making as a multicriteria problem that considers various points of view, such as financial, human resources, and environmental aspects.² Multicriteria decision analysis aims to assist a decision maker in choosing the best alternative when multiple criteria conflict and compete with each other. Most commonly used decision-aiding methods, such as outranking methods and the analytical hierarchy process, are based on multicriteria aggregation procedures. Outranking methods determine which alternatives are preferred to others by systematically comparing possible alternatives on each criterion. The analytical hierarchy process structures multiple criteria into a hierarchy and calculates each criterion's score as a weighted sum of its subcriteria.

We can find multicriteria decision problems in other fields as well. The marketing research literature often regards buying a product as a multicriteria decision problem. For example, when we purchase a car, we consider multiple attributes, such as price, brand, and color. Marketing researchers commonly use the conjoint model technique for solving multicriteria problems.³ This model determines the importance weights of product attributes and their values. Customer preferences for the product can then be calculated as a linear combination of weighted values.

Similarly, certain electronic market mechanisms, such as multiattribute auctions, also use multicriteria information. Multiattribute auctions are typically used in procurement settings and enable participants to negotiate not only price but also other attributes such as quality level, style, and delivery date. Martin Bichler has demonstrated several advantages of multiattribute auctions over their single-attribute (say, price only) counterparts, including improved overall utility and suitability for various application domains.⁴

However, these multicriteria problems typically do not apply to personalization and recommendation settings. Instead, they involve finding solutions or items that are optimal in general (that is, optimal with respect to all users) and do not explicitly consider differences in individual user preferences. However, multicriteria rating problems have recently shown up in the recommender systems research literature as an important next-generation issue.⁵ In recommender systems literature, the roots of multicriteria ratings could be traced to methods for incorporating content-based features into collaborative recommendation techniques. For example, such recommender systems could identify favorite content attributes, such as "comedy" movies, based on the content analysis of previously rated items. This system could then recommend items to a user based not only on ratings of similar users but also on these favorite content attributes.⁶ However, the users could submit just a single rating for each item; they could not specify their individualized feedback about a specific movie component or aspect, such as its visual effects.

In addition, Francesco Ricci and Hannes Werthner developed a recommender system using case-based reasoning techniques to personalize travel.⁷ The system performs recommendations by ranking and aggregating elementary items (locations, activities, services) according to the user's preferences and a repository of

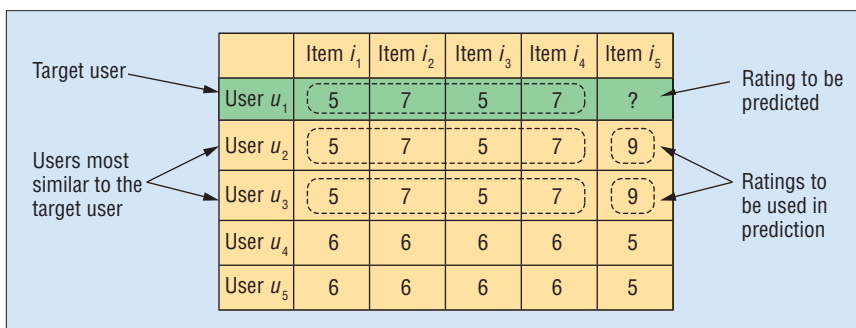


Figure 1. Collaborative filtering in a single-rating setting.

system determines who the "true peers" (or "nearest neighbors") of u are, the more accurate the rating prediction should be. Traditional 2D collaborative filtering calculates the similarity between users u and u' on the basis of how similar their ratings are for the movies they have both seen.

Figure 1 illustrates this estimation process with a simple example. Assume that we have five users u_1, \dots, u_5 and five movies i_1, \dots, i_5 , and suppose that the recommender system needs to estimate how much the target user u_1 would like movie i_5 . Furthermore, as figure 1 indicates, suppose that the system knows all other ratings of different users to different movies. The traditional collaborative-filtering approach finds the users that are closest to u_1 who have seen movie i_5 . In this case, figure 1 indicates that u_2 and u_3 are perfect matches for user u_1 because all of them rated the common movies exactly the same. Since both u_2 and u_3 rate movie i_5 as 9, the system will predict the value of target rating $R(u_1, i_5)$ as 9.

Now let's consider the same scenario but in a multicriteria setting. Specifically, let's assume the same five users u_1, \dots, u_5 and five movies

the recommender system a single rating (between 1 and 13) for each movie they have seen. Suppose also that this recommender system is using a traditional user-based collaborative-filtering approach for rating prediction, as we described earlier. In this case, according to equation 3, the system would estimate any rating that user u would give to yet-unseen movie i according to how users u' who are similar to target user u rated movie i . In other words, the system calculates unknown rating $R(u, i)$ on the basis of ratings $R(u', i)$. So, the more accurately the

past travel. Although these techniques do not use multicriteria ratings per se, the recommendation process does take into account multiple criteria, and the optimization is performed over a multidimensional solution space.

Furthermore, some research exists on how to filter recommendations on the basis of an item's content information. For example, J. Ben Schafer implements a metarecommendation system that lets users indicate their preference for each content attribute (for example, movie genre, Motion Picture Association of America rating, or film length) and rate the importance of these attributes.⁸ Users can indicate that they want only comedy movies and that it is the most important condition for recommendations. The users' requirements will filter the potential recommendations toward what they really want. However, this does not represent a multicriteria rating environment because the users are specifying general filtering requirements for all movies (such as specifying the preferred value and weight for movie genre attribute). Similarly, Wei-Po Lee, Chih-Hung Liu, and Cheng-Che Lu also obtain the importance weights of content attributes directly from the user.⁹ They use each attribute's rank to compare the items, but the value or rank of each attribute is assumed to be the same for all users. In contrast to the systems of Schafer⁸ and of Lee and his colleagues,⁹ multicriteria rating environments would let users specify subjective ratings for various components of *individual* items. For example, a user would be able to rate the visual effects component for *Star Wars*, which the system could then leverage for prediction and personalization purposes.

In summary, although the personalization literature includes several approaches that are somewhat related to the issue of incorporating and leveraging multicriteria ratings in recommender systems, it is fair to say that this issue has remained largely unexplored.

References

1. R.B. Statnikov and J.B. Matusov, *Multicriteria Optimization and Eng.*, Chapman & Hall, 1995.
2. J. Figueria, S. Greco, and M. Ehrgott, *Multiple Criteria Decision Analysis: State of the Art Surveys*, Springer, 2005.
3. P.E. Green, A.M. Krieger, and Y. Wind, "Thirty Years of Conjoint Analysis: Reflections and Prospects," *Interfaces*, vol. 31, no. 3, 2001, pp. 56–73.
4. M. Bichler, "An Experimental Analysis of Multi-Attribute Auctions," *Decision Support Systems*, vol. 29, no. 10, 2000, pp. 249–268.
5. G. Adomavicius and A. Tuzhilin, "Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions," *IEEE Trans. Knowledge and Data Eng.* vol. 17, no. 6, 2005, pp. 734–749.
6. M. Balabanovic and Y. Shoham, "Fab: Content-Based, Collaborative Recommendation," *Comm. ACM*, vol. 40, no. 3, 1997, pp. 66–72.
7. F. Ricci and H. Werthner, "Case-Based Querying for Travel Planning Recommendation," *Information Technology and Tourism*, vol. 4, nos. 3–4, 2002, pp. 215–226.
8. J. B. Schafer, "DynamicLens: A Dynamic User-Interface for a Meta-Recommendation Systems," *Proc. Beyond Personalization 2005: A Workshop on the Next Stage of Recommender Systems Research*, 2005, pp. 72–76; www.cs.umn.edu/Research/GroupLens/beyond2005.
9. W. Lee, C. Liu, and C. Lu, "Intelligent Agent-Based Systems for Personalized Recommendations in Internet Commerce," *Expert Systems with Applications*, vol. 22, no. 4, 2002, pp. 275–184.

i_1, \dots, i_5 , an unknown rating $R(u_1, i_5)$ that must be predicted, and known overall ratings of all users to different movies that are exactly the same as in figure 1. In addition, however, assume that the system asks each user to provide the feedback about the movie on four specific criteria—story, acting, direction, and visuals—as movie review sites such as Yahoo! Movies (<http://movies.yahoo.com>) do. Finally, assume that the overall rating in this case is a simple average of the four individual criteria ratings.

Following the idea behind the standard collaborative-filtering approach, the recommender system should predict $R(u_1, i_5)$ by finding the users who are closest to u_1 and who have seen i_5 . However, the additional information available in the multicriteria ratings shown in figure 2 makes it clear that u_2 and u_3 are quite different in their tastes and preferences from u_1 , even though their overall ratings for each movie match perfectly. In particular, u_1 disliked the movie aspects (story and acting) that u_2 and u_3 liked and liked the aspects (direction and visuals) they disliked. However,

	Item i_1	Item i_2	Item i_3	Item i_4	Item i_5
Target user User u_1	5 _{2,2,8,8}	7 _{5,5,9,9}	5 _{2,2,8,8}	7 _{5,5,9,9}	?
Users most similar to the target user User u_2	5 _{8,8,2,2}	7 _{9,9,5,5}	5 _{8,8,2,2}	7 _{9,9,5,5}	9
User u_3	5 _{8,8,2,2}	7 _{9,9,5,5}	5 _{8,8,2,2}	7 _{9,9,5,5}	9
User u_4	6 _{3,3,9,9}	6 _{4,4,8,8}	6 _{3,3,9,9}	6 _{4,4,8,8}	5
User u_5	6 _{3,3,9,9}	6 _{4,4,8,8}	6 _{3,3,9,9}	6 _{4,4,8,8}	5

Figure 2. Collaborative filtering in a multicriteria setting, where the overall rating for each movie i is a simple average of four rating criteria: story, acting, direction, and visuals.

recommender systems that are based on single-criterion ratings would hide this information in the aggregated rating. As it does in this example, the aggregation can lead to inaccurate insights about the true similarity between user preferences.

Users u_4 and u_5 seem to be much better matches for user u_1 in this example. Not only their overall ratings are similar but also are their

preferences for different movie aspects. Both u_4 and u_5 rate movie i_5 as 5, so the system would predict a value of 5 for the target rating $R(u_1, i_5)$. This outcome is very different from the one obtained in a single-rating scenario.

In summary, the overall rating that users give to an item provides the information regarding *how much* they like the item, and multicriteria ratings provide some insights regarding *why* they like it. Therefore, multicriteria ratings enable more accurate estimates of the similarity between two users. Accordingly, we propose to extend the standard collaborative-filtering algorithm to include multicriteria ratings. Specifically, we propose including multicriteria rating information in the similarity calculation between two different users, $\text{sim}(u, u')$, or two different items, $\text{sim}(i, i')$. Then, given the newly calculated similarity, a recommender system can use the weighted sum or adjusted weighted sum to predict unknown ratings, just as it does with a standard collaborative-filtering algorithm—that is, using equations 2, 3, or 5.

We will describe two different approaches to leveraging multicriteria ratings in the similarity computation. These approaches change only the similarity function in the traditional collaborative-filtering technique to reflect multicriteria rating information.

Aggregating traditional similarities from individual criteria. This approach can use any standard similarity metric, such as cosine-based (equation 4), and calculate the similarity between users (or items) on the basis of each individual criterion. Let's assume that each rating user u gives to item i consists of an overall rating r_0 and k multicriteria ratings r_1, \dots, r_k :

$$R(u, i) = (r_0, r_1, \dots, r_k). \quad (7)$$

Then, we can obtain $k + 1$ different similarity estimations by using some standard metric to measure similarity between users u and u' : $\text{sim}_0(u, u')$ represents the similarity between u and u' based on the overall rating; $\text{sim}_1(u, u')$ represents the similarity based on the first criterion rating; $\text{sim}_2(u, u')$, similarity based on the second criterion rating; and so on. We can then compute the overall similarity by aggregating the individual similarities in several ways, including *average similarity*—that is, by averaging all individual similarities,

$$\text{sim}_{\text{avg}}(u, u') = \frac{1}{k+1} \sum_{i=0}^k \text{sim}_i(u, u') \quad (8)$$

and *worst-case similarity*—that is, by using the smallest of similarities,

$$\text{sim}_{\text{min}}(u, u') = \min_{i=0, \dots, k} \text{sim}_i(u, u') \quad (9)$$

Calculating similarity using multidimensional distance metrics. In a multicriteria rating scenario, each rating $R(u, i) = (r_0, r_1, \dots, r_k)$ represents a point in the $k + 1$ -dimensional space. So, one natural approach to computing similarity between different users is to use multidimensional distance metrics. Such metrics are easy to understand and straightforward to implement. Note that the metrics of distance and similarity are inversely related: the smaller the distance between two users, the higher the similarity. We calculate the similarity between two users in three steps.

First, we must calculate the distance between two users' ratings for the same item—that is, $d_{\text{rating}}(R(u, i), R(u', i))$, where $R(u, i) = (r_0,$

$r_1, \dots, r_k)$ and $R(u', i) = (r'_0, r'_1, \dots, r'_k)$. For this purpose, we can use any of the standard multidimensional distance metrics:

- Manhattan distance:

$$\sum_{i=0}^k |r_i - r'_i| \quad (10)$$

- Euclidean distance:

$$\sqrt{\sum_{i=0}^k |r_i - r'_i|^2} \quad (11)$$

- Chebyshev (or maximal value) distance:

$$\max_{i=0, \dots, k} |r_i - r'_i| \quad (12)$$

Second, the overall distance between two users u and u' is simply

$$d_{\text{user}}(u, u') = \frac{1}{|I(u, u')|} \sum_{i \in I(u, u')} d_{\text{rating}}(R(u, i), R(u', i)) \quad (13)$$

where $I(u, u')$ denotes the set of items that both u and u' have rated. In other words, the overall distance between two users u and u' is the average distance between their ratings for all their common items.

Finally, because the collaborative-filtering techniques operate with the metric of user similarity (and not user distance), and the distance and similarity are inversely related, we use the simple transformation between the two metrics:

$$\text{sim}(u, u') = \frac{1}{1 + d_{\text{user}}(u, u')} \quad (14)$$

Note that this definition of similarity has desirable range properties: the similarity will approach 0 as the distance between two users becomes larger, and it will be 1 if the distance is 0 (users are identical).

Aggregation-function-based approach

The approaches we have just described apply primarily to similarity-based recommenders, such as traditional collaborative-filtering systems. We now present an approach that is not limited to any specific recommendation algorithm. The intuition behind this approach comes from the assumption that multicriteria ratings represent user preferences for different components of an item, such as story, acting, direction, and visuals in the case of movies. So, an item's overall rating is not just another rating that is independent of others; rather, it serves as some aggregation function f of the item's multicriteria ratings:

$$r_0 = f(r_1, \dots, r_k) \quad (15)$$

In other words, this approach assumes that the overall rating has a certain relationship with the multicriteria ratings. For instance, in a movie recommendation application, the story criterion rating might have a very high priority—that is, movies with high story ratings are well liked overall by some users, regardless of other criteria ratings. So, if a system predicts that a movie's story rating will be high, it must also predict that the overall rating will be high in order to be accurate.

Figure 3 illustrates the proposed three-step approach to rating estimation.

Step 1: Predict multicriteria ratings. First, we decompose the k -dimensional multicriteria rating space into k single-rating recommendation problems, where each problem can be represented with a traditional $Users \times Items$ matrix (like the one in figure 1). In other words, instead of the multicriteria recommendation problem $R: Users \times Items \rightarrow R_0 \times R_1 \times \dots \times R_k$, we are dealing with k single-rating recommendation problems $R: Users \times Items \rightarrow R_i$ (where $i = 1, \dots, k$).

This approach provides a lot of flexibility. Unlike the similarity-based approaches, it can use any existing single-rating recommendation technique—collaborative, content-based, or hybrid—to estimate unknown ratings for each individual criterion.

Step 2: Learn the aggregation function. This step aims to estimate relationship f between the overall rating and the underlying multicriteria ratings of items, such that $r_0 = f(r_1, \dots, r_k)$. We can already predict the individual multicriteria ratings (step 1), but the ability to predict the overall rating of each item for each user is also useful in many situations. For example, with the overall rating for each item, a system can rank all items for each user and recommend only the most relevant items. To determine the most relevant items without an overall rating, the system would have to deal with a much more complex multicriteria optimization problem.⁵ Thus, finding the aggregation function is crucial for recommender systems, and there are several ways to obtain it:

- *Domain expertise.* On the basis of prior experience and domain knowledge, a domain expert can suggest an appropriate aggregation function. For example, the overall rating might be a simple average of the underlying multicriteria ratings for each item: $r_0 = (r_1 + \dots + r_k)/k$.
- *Statistical techniques,* including various linear and nonlinear regression-analysis techniques. For example, in linear regression, the aggregation function for the overall rating would be a linear combination of the multicriteria ratings—that is, $r_0 = w_1 r_1 + \dots + w_k r_k + c$ —where we can interpret weight w_i associated with criterion i as this criterion’s importance in determining the overall rating. We can estimate the weights w_i ($i = 1, \dots, k$) and constant c based on the set of known ratings.
- *Machine learning techniques.* We can also obtain the function using various sophisticated computational learning techniques such as artificial neural networks.⁶

Besides supporting different learning techniques, the aggregation function can also be of three different *scopes*: total, user-based, or item-based. In particular, f is a *total aggregation function* if it is used to predict all unknown ratings—for example, if the criteria weights w_i in a regression-based function are the same for all users and items. However, depending on the domain specifics, user-based or item-based aggregation functions can also be useful in some applications. For example, in a movie recommender system, u might consistently give greater weight to the “story” component of all movies, whereas u' might give significant weight to the “visuals” component. In this case, it would be advantageous for u to have his or her own *user-*

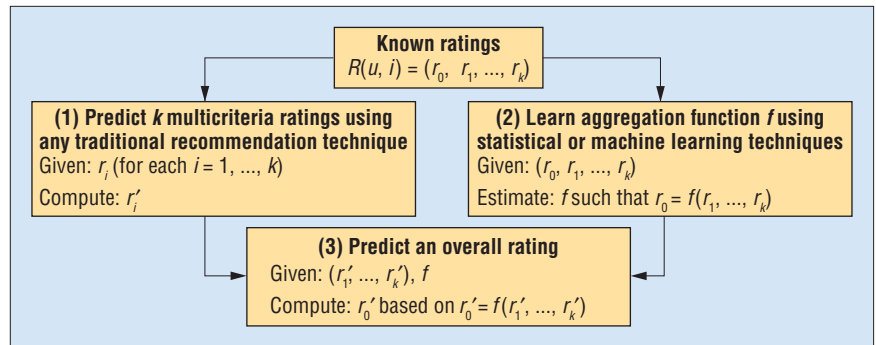


Figure 3. Overview of the aggregation-function-based approach.

based aggregation function f_{iu} , which the system would learn exclusively from u ’s known ratings (as opposed to all known ratings). Similarly, with the *item-based aggregation function* f_i , we would assume that each item i will have its own aggregation function that is based on all the ratings involving this item.

Various techniques are available for testing the fitness or accuracy of the learned aggregation functions. For example, in the case of linear regression, we can estimate the predictive power using its R-squared value. Or, more generally, we could use a standard n -fold cross-validation technique to estimate the predictive accuracy.⁶ This means we can restrict the use of user-based (or item-based) aggregation functions to those that exhibit sufficient predictive performance—for example, that exceed some prespecified threshold. The remaining users (or items) could use other techniques, such as the total aggregation function. As with every data-driven computational learning technique, this approach will work well in some application domains—for example, where users or items exhibit consistent preferences on each criterion. In other domains, other techniques might prove more advantageous.

Step 3: Predict overall ratings. Finally, we compute each unknown overall rating r'_0 directly by using the multicriteria ratings estimated in step 1 and function f estimated in step 2:

$$r'_0 = f(r'_1, \dots, r'_k)$$

Experimental results

To evaluate the proposed approaches, we collected a set of user-submitted movie ratings from the Yahoo! Movies Web site (<http://movies.yahoo.com>) for several hundred randomly chosen movies from the last decade. When users submit movie ratings to Yahoo! Movies, in addition to the overall rating, they are asked to provide four criteria information for each movie: story, acting, direction, and visuals. All ratings have 13 possible values and are based on a standard grading scale from A+ to F; for our analysis, we changed the grades to numerical values from 13 to 1. In the data preprocessing stage, we invoked two constraints on the data set to ensure that it was not extremely sparse and had sufficient data for rating prediction: that each user rated at least 10 movies and that each movie had at least 10 user ratings.

The resulting data set included 155 users, 50 movies, and 2,216 known ratings in total. The data set’s sparsity is 28.6 percent—that is, 28.6 percent of ratings are known. Each user has rated 14.3 movies on average, and the average number of common movies between two users is 5.2. On average, each movie has been rated by 44.3 users, and

Table 1. Experimental results of several recommendation approaches.

Recommendation approach:		Precision in top 3 (%)	Precision in top 5 (%)	Precision in top 7 (%)
user-based CF				
Neighborhood size: all users	standard CF	70.7*	68.7	69.0
	cos-min	70.7**	68.8	69.1
	Chebyshev	74.5†	70.3	70.5
	total-reg	71.5	70.9	70.4
	movie-reg95	71.8	74.0‡	75.3
Neighborhood size: three users	standard CF	64.9	64.9	66.3
	cos-min	67.1	67.1	67.8
	Chebyshev	66.2	65.5	64.6
	total-reg	65.2	66.6	66.5
	movie-reg95	69.0	70.7	72.2

*Shaded cells represent the performance of the standard collaborative-filtering baseline approach.
 ** Roman font indicates precision values that represent 0–1% improvement over the baseline approach.
 †Italic font indicates precision values that represent 1–4% improvement over the baseline approach.
 ‡Bold font indicates precision values that represent >4% improvement over the baseline approach.

the average number of common users between two movies is 13.6. The average rating on each criterion is approximately 9 (or “B”).

Furthermore, to obtain reliable results with a relatively small amount of data, we used a standard 10-fold cross-validation technique,⁶ where we randomly divided the data set into 10 disjoint subsets. We used nine-tenths of the data for training, and the remaining one-tenth to test rating prediction. Then we repeated this process 10 times (each time with a different test dataset) and performed the evaluation on all predicted ratings.

The research literature proposes numerous metrics for evaluating recommender system performance.⁷ These include statistical accuracy metrics, such as root mean squared error, as well as decision-support measures that determine how well the recommendation algorithm can predict items the user would rate highly. Example decision-support metrics include *precision* (percentage of truly “high” ratings among those that the system predicted would be high), *recall* (the percentage of correctly predicted high ratings among all the ratings known to be high), and *F-measure*, which is a harmonic mean of precision and recall.⁷

Here, we focus on a popular variation of the precision metric—namely, *precision-in-top-N*. This metric represents the percentage of truly high overall ratings among those that the system predicted would be the *N* most relevant items for each user. We chose this metric because of its practicality, given that many users in real-life personalization and recommendation applications are typically interested in looking at only a few highest-ranked item recommendations.

Because precision-related metrics measure the frequency with which a system correctly predicts a high rank for an item that is truly high-ranked, we needed to define what “high-ranked” means in our application. In other words, we had to define every rating on a binary scale, as high-ranked or not high-ranked. Because Yahoo! Movies’ rating scale was not binary, we translated the overall movie ratings into a binary scale by treating the ratings greater than 10.5 (A+, A, A–) as high-ranked and ratings less than 10.5 as not high-ranked. The 10.5 threshold assumes that users would want to focus on recommendations about movies that are most relevant to them (movies they would rate as A+, A, A–) and that correctness is therefore most desirable in recommendations for such movies.

In our data set, 35.6 percent of the overall ratings were above 10.5, which means that simply recommending items at random could achieve the precision level of 35.6 percent. Any recommender system that does not achieve 35.6 percent precision would be worse than a random guess and, therefore, essentially useless.

Table 1 summarizes the performance of the following five recommendation approaches on the movie data set:

- *Standard CF*—a traditional single-rating user-based collaborative-filtering approach, which uses adjusted weighted sum and the cosine similarity metric, as described in equations 3 and 4. We used this approach as a baseline to compare a single-rating system to the multicriteria recommendation approaches.
- Two similarity-based techniques implemented with the traditional user-based collaborative-filtering approach: *cos-min*—a technique that aggregates traditional cosine-based similarities for each individual rating, and *Chebyshev*—a technique that uses the Chebyshev multidimensional distance metric.
- Two aggregation-function-based techniques that use the traditional user-based collaborative-filtering approach to estimate individual multicriteria ratings: *total-reg*—a total aggregation function that is based on linear regression, and *movie-reg95*—an item-based aggregation function that is generated separately for each movie and restricted to movies that have the best regression fit—specifically, where R-squared ≥ 95 percent.

We used the standard user-based collaborative-filtering approach as an integral part of every technique to minimize the nonessential differences between techniques as much as possible and, thus, to maximize the possibility that any performance differences between the standard CF and multicriteria recommender systems are due to the newly introduced multicriteria rating information.

For the sake of completeness, table 1 includes results for different collaborative-filtering neighborhood sizes (all users versus the three most similar users) and for different precision-in-top-*N* levels (*N* = 3, 5, and 7). The shaded cells in the table represent the performance of the standard CF baseline approach. As the table shows,

nearly every multicriteria technique performed at least as well as or better than the baseline: precision values in regular font represent improvement between 0 and 1 percent; italic font, from 1 to 4 percent; and boldface, greater than 4 percent.

For further comparison, we calculated the precision-in-top- N metric for a simple popularity-based recommendation approach, in which the system recommends N movies ($N = 3, 5,$ and 7) that are most liked by all other users, as indicated by each movie's average rating. The precision-in-top- N results for this simple approach were 61.3 percent for $N = 3$, 53.3 percent for $N = 5$, and 46.4 percent for $N = 7$. These values all exceed the "random guess" threshold of 35.6 percent but fall short of the performance achieved with collaborative-filtering techniques.

Among other notable results:

- We tried *precision-in-top-1* measures (as opposed to the top 3, 5, and 7 measures in table 1) for various neighborhood sizes. Similarity-based techniques (such as Chebyshev and cos-min) performed best, typically exceeding both the baseline approach and the aggregation-function-based approaches by 2 to 6 percent.
- We also tried movie-based (as opposed to user-based) collaborative filtering. In this case, total-reg performed the best of all the techniques for various neighborhood sizes and typically outperformed the baseline approach by 1 to 5 percent.
- The performance differences between multicriteria recommendation techniques and the baseline approach are even larger in a sparser environment. For instance, we tested our recommendation algorithms on a sparser data set, which we obtained from our initial data set by randomly removing about one-third of its ratings (specifically, 700 ratings out of 2,216). The new data set's sparsity was 19.6 percent. Most multicriteria recommendation algorithms (including Chebyshev and aggregation-function-based techniques) outperformed the baseline approach by 5 to 10 percent on this sparser data set.
- Combining similarity-based and aggregation-function-based multicriteria recommendation techniques can sometimes improve the predictive performance. This result is generally consistent with similar findings in recommender systems literature about the advantages of combining different types of recommender systems. For example, it has been widely reported that combining content-based and collaborative systems can improve recommendation accuracy.

As with most recommender systems and, more generally, computational learning techniques, the performance of a specific technique is highly domain-dependent. In other words, performance depends significantly on the underlying data's characteristics. So, although we expect the proposed techniques to do well in a variety of application domains, we do not expect them to outperform traditional single-criterion techniques in all domains where multicriteria information exists. For example, these techniques cannot be expected to perform well in domains where multicriteria ratings do not carry meaningful information or where no inherent relationship exists between the overall rating and the multicriteria ratings for the users or items.

The recommender systems field has made significant progress over the past few years with many new techniques proposed and new systems developed. However, modern systems still require significant improvements to provide better recommendations and be

viable in more complex personalization applications. The ability to leverage multicriteria rating information constitutes one such improvement. We expect that the two recommendation approaches we have proposed—similarity-based and aggregation-function-based—will be useful in many personalization-related application domains. Our experimental results on a real-world data set confirm that, when available, multicriteria ratings can be successfully leveraged to improve recommendation accuracy. We believe that this article is just the first step in studying multicriteria recommender systems and that significant additional work is needed to further explore this issue. ■

Acknowledgments

The US National Science Foundation supported part of the research reported in this article under grant IIS-0546443.

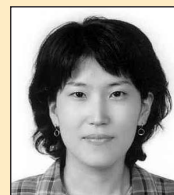
References

1. M. Balabanovic and Y. Shoham, "Fab: Content-Based, Collaborative Recommendation," *Comm. ACM*, vol. 40, no. 3, 1997, pp. 66–72.
2. G. Adomavicius and A. Tuzhilin, "Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions," *IEEE Trans. Knowledge and Data Engineering*, vol. 17, no. 6, 2005, pp. 734–749.
3. J.S. Breese, D. Heckerman, and C. Kadie, "Empirical Analysis of Predictive Algorithms for Collaborative Filtering," *Proc. 14th Conf. Uncertainty in Artificial Intelligence*, Morgan Kaufmann, 1998, pp. 43–52.
4. B. Sarwar et al., "Item-Based Collaborative Filtering Recommendation Algorithms," *Proc. 10th Int'l WWW Conf.*, ACM Press, 2001, pp. 285–295.
5. R.B. Statnikov and J.B. Matusov, *Multicriteria Optimization and Engineering*, Chapman & Hall, 1995.
6. T. Mitchell, *Machine Learning*, McGraw-Hill, 1997.
7. J.L. Herlocker et al., "Evaluating Collaborative Filtering Recommender Systems," *ACM Trans. Information Systems*, vol. 22, no. 1, 2004, pp. 5–53.

The Authors



Gediminas Adomavicius is a Digital Technology Center assistant professor of information and decision sciences at the Carlson School of Management, University of Minnesota. His research interests include personalization technologies, recommender systems, data mining, and electronic market mechanisms. He received his PhD in computer science from New York University. He's a member of the IEEE, ACM, and Association for Information Systems. Contact him at the Dept. of Information and Decision Sciences, Carlson School of Management, Univ. of Minnesota, 321 19th Ave. South, Minneapolis, MN 55455; gedas@umn.edu.



YoungOk Kwon is a PhD student in information and decision sciences at the Carlson School of Management, University of Minnesota. Her research interests include recommender systems, data mining, and business intelligence. She received her MBA from Seoul National University, Korea. Contact her at the Dept. of Information and Decision Sciences, Carlson School of Management, Univ. of Minnesota, 321 19th Ave. South, Minneapolis, MN 55455; ykwon@som.umn.edu.