the authority scores are each equal to 1. Higher values of the score indicate better quality. The hub and authority scores are related to one another in the following way:

$$h(i) = \sum_{j:(i,j)\in A} a(j) \quad \forall i \in S \tag{18.10}$$

$$a(i) = \sum_{j:(j,i)\in A} h(j) \quad \forall i \in S. \tag{18.11}$$

The basic idea is to reward hubs for pointing to good authorities and reward authorities for being pointed to by good hubs. It is easy to see that the aforementioned system of equations reinforces this mutually enhancing relationship. This is a linear system of equations that can be solved using an iterative method. The algorithm starts by initializing $h^0(i) = a^0(i) = 1/\sqrt{|S|}$. Let $h^t(i)$ and $a^t(i)$ denote the hub and authority scores of the ith node, respectively, at the end of the tth iteration. For each $t \geq 0$, the algorithm executes the following iterative steps in the $(t+1)$th iteration:

for each $i \in S$ set $a^{t+1}(i) \Leftarrow \sum_{j:(j,i)\in A} h^t(j)$;
for each $i \in S$ set $h^{t+1}(i) \Leftarrow \sum_{j:(i,j)\in A} a^{t+1}(j)$;
Normalize L_2-norm of each of hub and authority vectors to 1;

For hub-vector $\overline{h} = [h(1) \ldots h(n)]^T$ and authority-vector $\overline{a} = [a(1) \ldots a(n)]^T$, the updates can be expressed as $\overline{a} = A^T\overline{h}$ and $\overline{h} = A\overline{a}$, respectively, when the edge set A is treated as an $|S| \times |S|$ adjacency matrix. The iteration is repeated to convergence. It can be shown that the hub vector $\overline{h}$ and the authority vector $\overline{a}$ converge in directions proportional to the dominant eigenvectors of AA^T and A^TA (see Exercise 6), respectively. This is because the relevant pair of updates can be shown to be equivalent to power-iteration updates of AA^T and A^TA, respectively.

18.5 Recommender Systems

Ever since the popularization of web-based transactions, it has become increasingly easy to collect data about user buying behaviors. This data includes information about user profiles, interests, browsing behavior, buying behavior, and ratings about various items. It is natural to leverage such data to make recommendations to customers about possible buying interests.

In the recommendation problem, the user–item pairs have *utility values* associated with them. Thus, for n users and d items, this results in an $n \times d$ matrix D of utility values. This is also referred to as the *utility-matrix*. The utility value for a user-item pair could correspond to either the buying behavior or the ratings of the user for the item. Typically, a small subset of the utility values are specified in the form of either customer buying behavior or ratings. It is desirable to use these specified values to make recommendations. The nature of the utility matrix has a significant influence on the choice of recommendation algorithm:

1. *Positive preferences only:* In this case, the specified utility matrix only contains positive preferences. For example, a specification of a "like" option on a social networking site, the browsing of an item at an online site, or the buying of a specified quantity of an item, corresponds to a positive preference. Thus, the utility matrix is sparse, with a prespecified set of positive preferences. For example, the utility matrix may contain the raw quantities of the item bought by each user, a normalized mathematical function of the quantities, or a weighted function of buying and browsing behavior. These

functions are typically specified heuristically by the analyst in an application-specific way. Entries that correspond to items not bought or browsed by the user may remain unspecified.

2. *Positive and negative preferences (ratings):* In this case, the user specifies the ratings that represent their like or dislike for the item. The incorporation of user dislike in the analysis is significant because it makes the problem more complex and often requires some changes to the underlying algorithms.

An example of a ratings-based utility matrix is illustrated in Fig. 18.4a, and an example of a positive-preference utility matrix is illustrated in Fig. 18.4b. In this case, there are six users, labeled $U_1 \ldots U_6$, and six movies with specified titles. Higher ratings indicate more positive feedback in Fig. 18.4a. The missing entries correspond to unspecified preferences in both cases. This difference significantly changes the algorithms used in the two cases. In particular, the two matrices in Fig. 18.4 have the same specified entries, but they provide very different insights. For example, the users U_1 and U_3 are very different in Fig. 18.4a because they have very different ratings for their commonly specified entries. On the other hand, these users would be considered very similar in Fig. 18.4b because these users have expressed a positive preference for the same items. The ratings-based utility provides a way for users to express negative preferences for items. For example, user U_1 does not like the movie *Gladiator* in Fig. 18.4a. There is no mechanism to specify this in the positive-preference utility matrix of Fig. 18.4b beyond a relatively ambiguous missing entry. In other words, the matrix in Fig. 18.4b is less expressive. While Fig. 18.4b provides an example of a binary matrix, it is possible for the nonzero entries to be arbitrary positive values. For example, they could correspond to the quantities of items bought by the different users.

This difference has an impact on the types of algorithms that are used in the two cases. Allowing for positive and negative preferences generally makes the problem harder. From a data collection point of view, it is also harder to infer negative preferences when they are inferred from customer behavior rather than ratings. Recommendations can also be enhanced with the use of content in the user and item representations.

1. *Content-based recommendations:* In this case, the users and items are both associated with feature-based descriptions. For example, item profiles can be determined by using the text of the item description. A user might also have explicitly specified their interests in a profile. Alternatively, their profile can be inferred from their buying or browsing behavior.

2. *Collaborative filtering:* Collaborative filtering, as the name implies, is the leveraging of the user preferences in the form of ratings or buying behavior in a "collaborative" way, for the benefit of all users. Specifically, the utility matrix is used to determine either relevant users for specific items, or relevant items for specific users in the recommendation process. A key intermediate step in this approach is the determination of similar groups of items and users. The patterns in these peer groups provide the collaborative knowledge needed in the recommendation process.

The two models are not exclusive. It is often possible to combine content-based methods with collaborative filtering methods to create a combined preference score. Collaborative filtering methods are generally among the more commonly used models and will therefore be discussed in greater detail in this section.

It is important to understand that the utility matrices used in collaborative filtering algorithms are extremely large and sparse. It is not uncommon for the values of n and d in

	GLADIATOR	GODFATHER	BEN-HUR	GOODFELLAS	SCARFACE	SPARTACUS
U_1	1			5		2
U_2		5			4	
U_3	5	3		1		
U_4			3			4
U_5				3	5	
U_6	5		4			

(a) Ratings-based utility

	GLADIATOR	GODFATHER	BEN-HUR	GOODFELLAS	SCARFACE	SPARTACUS
U_1	1			1		1
U_2		1			1	
U_3	1	1		1		
U_4			1			1
U_5				1	1	
U_6	1		1			

(b) Positive-preference utility

Figure 18.4: Examples of utility matrices.

the $n \times d$ utility matrix to exceed 10^5. The matrix is also extremely *sparse*. For example, in a movie data set, a typical user may have specified no more than 10 ratings, out of a universe of more than 10^5 movies.

At a basic level, collaborative filtering can be viewed as a missing-value estimation or matrix completion problem, in which an incomplete $n \times d$ utility matrix is specified, and it is desired to estimate the missing values. As discussed in the bibliographic notes, many methods exist in the traditional statistics literature on missing-value estimation. However, collaborative filtering problems present a particularly challenging special case in terms of data size and sparsity.

18.5.1 Content-Based Recommendations

In content-based recommendations, the user is associated with a set of documents that describe his or her interests. Multiple documents may be associated with a user corresponding to his or her specified demographic profile, specified interests at registration time, the product description of the items bought, and so on. These documents can then be aggregated into a single textual content-based profile of the user in a vector space representation.

The items are also associated with textual descriptions. When the textual descriptions of the items match the user profile, this can be viewed as an indicator of similarity. When no utility matrix is available, the content-based recommendation method uses a simple k-nearest neighbor approach. The top-k items are found that are closest to the user textual profile. The cosine similarity with tf-idf can be used, as discussed in Chap. 13.

On the other hand, when a utility matrix is available, the problem of finding the most relevant items for a particular user can be viewed as a traditional classification problem. For each user, we have a set of *training* documents representing the descriptions of the items for which that user has specified utilities. The labels represent the utility values. The descriptions of the remaining items for that user can be viewed as the test documents for classification. When the utility matrix contains numeric ratings, the class variables are

numeric. The regression methods discussed in Sect. 11.5 of Chap. 11 may be used in this case. Logistic and ordered probit regression are particularly popular. In cases where only positive preferences (rather than ratings) are available in the utility matrix, all the specified utility entries correspond to positive examples for the item. The classification is then performed only on the remaining test documents. One challenge is that only a small number of positive training examples are specified, and the remaining examples are unlabeled. In such cases, specialized classification methods using only positive and unlabeled methods may be used. Refer to the bibliographic notes of Chap. 11. Content-based methods have the advantage that they do not even require a utility matrix and leverage domain-specific content information. On the other hand, content information biases the recommendation towards items described by similar keywords to what the user has seen in the past. Collaborative filtering methods work directly with the utility matrix, and can therefore avoid such biases.

18.5.2 Neighborhood-Based Methods for Collaborative Filtering

The basic idea in neighborhood-based methods is to use either user–user similarity, or item–item similarity to make recommendations from a ratings matrix.

18.5.2.1 User-Based Similarity with Ratings

In this case, the top-k similar users to each user are determined with the use of a similarity function. Thus, for the target user i, its similarity to all the other users is computed. Therefore, a similarity function needs to be defined between users. In the case of a ratings-based matrix, the similarity computation is tricky because different users may have different scales of ratings. One user may be biased towards liking most items, and another user may be biased toward not liking most of the items. Furthermore, different users may have rated different items. One measure that captures the similarity between the rating vectors of two users is the Pearson correlation coefficient. Let $\overline{X} = (x_1 \ldots x_s)$ and $\overline{Y} = (y_1 \ldots y_s)$ be the common (specified) ratings between a pair of users, with means $\hat{x} = \sum_{i=1}^{s} x_i/s$ and $\hat{y} = \sum_{i=1}^{s} y_i/s$, respectively. Alternatively, the mean rating of a user is computed by averaging over all her specified ratings rather than using only co-rated items by the pair of users at hand. This alternative way of computing the mean is more common, and it can significantly affect the pairwise Pearson computation. Then, the Pearson correlation coefficient between the two users is defined as follows:

$$\text{Pearson}(\overline{X}, \overline{Y}) = \frac{\sum_{i=1}^{s}(x_i - \hat{x}) \cdot (y_i - \hat{y})}{\sqrt{\sum_{i=1}^{s}(x_i - \hat{x})^2} \cdot \sqrt{\sum_{i=1}^{s}(y_i - \hat{y})^2}}. \tag{18.12}$$

The Pearson coefficient is computed between the target user and all the other users. The peer group of the target user is defined as the top-k users with the highest Pearson coefficient of correlation with her. Users with very low or negative correlations are also removed from the peer group. The average ratings of each of the (specified) items of this peer group are returned as the recommended ratings. To achieve greater robustness, it is also possible to weight each rating with the Pearson correlation coefficient of its owner while computing the average. This weighted average rating can provide a prediction for the target user. The items with the highest predicted ratings are recommended to the user.

The main problem with this approach is that different users may provide ratings on different scales. One user may rate all items highly, whereas another user may rate all items negatively. The raw ratings, therefore, need to be normalized before determining the (weighted) average rating of the peer group. The normalized rating of a user is defined by

subtracting her mean rating from each of her ratings. As before, the weighted average of the normalized rating of an item in the peer group is determined as a *normalized* prediction. The mean rating of the target user is then added back to the normalized rating prediction to provide a *raw* rating prediction.

18.5.2.2 Item-Based Similarity with Ratings

The main conceptual difference from the user-based approach is that peer groups are constructed in terms of *items* rather than *users*. Therefore, similarities need to be computed between items (or columns in the ratings matrix). Before computing the similarities between the columns, the ratings matrix is normalized. As in the case of user-based ratings, the average of each row in the ratings matrix is subtracted from that row. Then, the cosine similarity between the normalized ratings $\overline{U} = (u_1 \ldots u_s)$ and $\overline{V} = (v_1 \ldots v_s)$ of a pair of items (columns) defines the similarity between them:

$$\text{Cosine}(\overline{U}, \overline{V}) = \frac{\sum_{i=1}^{s} u_i \cdot v_i}{\sqrt{\sum_{i=1}^{s} u_i^2} \cdot \sqrt{\sum_{i=1}^{s} v_i^2}}. \tag{18.13}$$

This similarity is referred to as the *adjusted* cosine similarity, because the ratings are normalized before computing the similarity value.

Consider the case in which the rating of item j for user i needs to be determined. The first step is to determine the top-k most similar *items* to *item* j based on the aforementioned adjusted cosine similarity. Among the top-k matching items to item j, the ones for which user i has specified ratings are determined. The *weighted* average value of these (raw) ratings is reported as the predicted value. The weight of item r in this average is equal to the adjusted cosine similarity between item r and the target item j.

The basic idea is to leverage the user's *own* ratings in the final step of making the prediction. For example, in a movie recommendation system, the item peer group will typically be movies of a similar genre. The previous ratings history of the *same* user on such movies is a very reliable predictor of the interests of that user.

18.5.3 Graph-Based Methods

It is possible to use a random walk on the user-item graph, rather than the Pearson correlation coefficient, for defining neighborhoods. Such an approach is sometimes more effective for sparse ratings matrices. A bipartite *user-item* graph $G = (N_u \cup N_i, A)$ is constructed, where N_u is the set of nodes representing users, and N_i is the set of nodes representing items. An undirected edge exists in A between a user and an item for each nonzero entry in the utility matrix. For example, the user-item graph for both utility matrices of Fig. 18.4 is illustrated in Fig. 18.5. One can use either the personalized *PageRank* or the *SimRank* method to determine the k most similar users to a given user for user-based collaborative filtering. Similarly, one can use this method to determine the k most similar items to a given item for item-based collaborative filtering. The other steps of user-based collaborative filtering and item-based collaborative filtering remain the same.

A more general approach is to view the problem as a positive and negative *link prediction problem* on the user-item graph. In such cases, the user-item graph is augmented with positive or negative weights on edges. The normalized rating of a user for an item, after subtracting the user-mean, can be viewed as either a positive or negative weight on the edge. For example, consider the graph constructed from the ratings matrix of Fig. 18.4(a). The edge between user U_1 and the item *Gladiator* would become a negative edge because

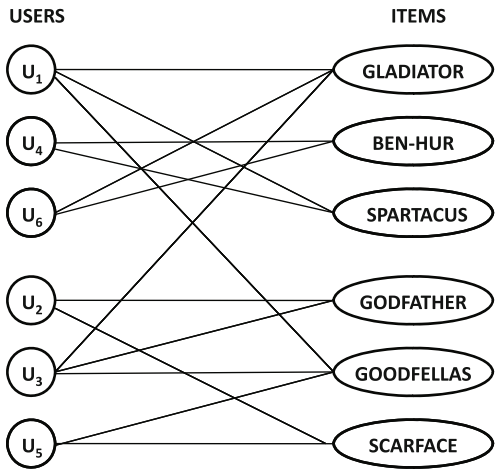


Figure 18.5: Preference graph for utility matrices of Fig. 18.4

U_1 clearly dislikes the movie *Gladiator*. The corresponding network would become a *signed* network. Therefore, the recommendation problem is that of predicting high positive weight edges between users and items in a signed network. A simpler version of the link-prediction problem with only positive links is discussed in Sect. 19.5 of Chap. 19. Refer to the bibliographic notes for link prediction methods with positive and negative links. The merit of the link prediction approach is that it can also leverage the available links between different users in a setting where they are connected by social network links. In such cases, the user-item graph no longer remains bipartite.

When users specify only positive preference values for items, the problem becomes simplified because most link prediction methods are designed for positive links. One can also use the random walks on the user-item graph to perform recommendations, rather than using it only to define neighborhoods. For example, in the case of Fig. 18.4b, the same user-item graph of Fig. 18.5 can be used in conjunction with a random-walk approach. This preference graph can be used to provide different types of recommendations:

1. The top ranking items for the user i can be determined by returning the item nodes with the largest *PageRank* in a random walk with restart at node i.

2. The top ranking users for the item j can be determined by returning the user nodes with the largest *PageRank* in a random walk with restart at node j.

The choice of the restart probability regulates the trade-off between the global popularity of the recommended item/user and the specificity of the recommendation to a particular user/item. For example, consider the case when items need to be recommended to user i. A low teleportation probability will favor the recommendation of popular items which are favored by many users. Increasing the teleportation probability will make the recommendation more specific to user i.

18.5.4 Clustering Methods

One weakness of neighborhood-based methods is the scale of the computation that needs to be performed. For *each user*, one typically has to perform computations that are proportional to *at least* the number of nonzero entries in the matrix. Furthermore, these computations need to be performed over *all* users to provide recommendations to different users.

This can be extremely slow. Therefore, a question arises, as to whether one can use clustering methods to speed up the computations. Clustering also helps address the issue of data sparsity to some extent.

Clustering methods are *exactly analogous* to neighborhood-based methods, except that the clustering is performed as a preprocessing step to define the peer groups. These peer groups are then used for making recommendations. The clusters can be defined either on users, or on items. Thus, they can be used to make either user-user similarity recommendations, or item-item similarity recommendations. For brevity, only the user-user recommendation approach is described here, although the item-item recommendation approach is exactly analogous. The clustering approach works as follows:

1. Cluster all the users into n_g groups of users using any clustering algorithm.
2. For any user i, compute the average (normalized) rating of the specified items in its cluster. Report these ratings for user i; after transforming back to the raw value.

The item–item recommendation approach is similar, except that the clustering is applied to the columns rather than the rows. The clusters define the groups of similar items (or implicitly pseudo-genres). The final step of computing the rating for a user-item combination is similar to the case of neighborhood-based methods. After the clustering has been performed, it is generally very efficient to determine all the ratings. It remains to be explained how the clustering is performed.

18.5.4.1 Adapting k-Means Clustering

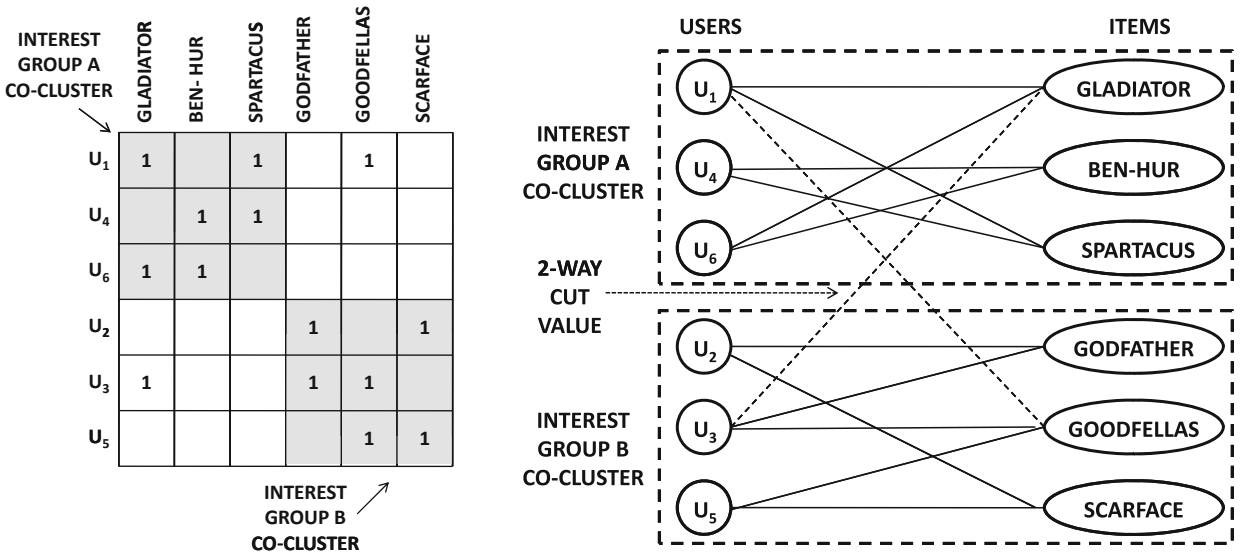
To cluster the ratings matrix, it is possible to adapt many of the clustering methods discussed in Chap. 6. However, it is important to adapt these methods to sparsely specified incomplete data sets. Methods such as k-means and Expectation Maximization may be used on the normalized ratings matrix. In the case of the k-means method, there are two major differences from the description of Chap. 6:

1. In an iteration of k-means, centroids are computed by averaging each dimension over the number of specified values in the cluster members. Furthermore, the centroid itself may not be fully specified.
2. The distance between a data point and a centroid is computed only over the specified dimensions in both. Furthermore, the distance is divided by the number of such dimensions in order to fairly compare different data points.

The ratings matrix should be normalized before applying the clustering method.

18.5.4.2 Adapting Co-Clustering

The co-clustering approach is described in Sect. 13.3.3.1 of Chap. 13. Co-clustering is well suited to discovery of neighborhood sets of users and items in sparse matrices. The specified entries are treated as 1s and the unspecified entries are treated as 0s for co-clustering. An example of the co-clustering approach, as applied to the utility matrix of Fig. 18.4b, is illustrated in Fig. 18.6a. In this case, only a 2-way co-clustering is shown for simplicity. The co-clustering approach cleanly partitions the users and items into groups with a clear correspondence to each other. Therefore, user-neighborhoods and item-neighborhoods are discovered simultaneously. After the neighborhoods have been defined, the aforementioned



(a) Co-cluster (b) User-item graph

Figure 18.6: Co-clustering of user-item graph

user-based methods and item-based methods can be used to make predictions for the missing entries.

The co-clustering approach also has a nice interpretation in terms of the user-item graph. Let $G = (N_u \cup N_i, A)$ denote the preference graph, where N_u is the set of nodes representing users, and N_i is the set of nodes representing items. An undirected edge exists in A for each nonzero entry of the utility matrix. Then the co-cluster is a clustering of this graph structure. The corresponding 2-way graph partition is illustrated in Fig. 18.6b. Because of this interpretation in terms of user-item graphs, the approach is able to exploit item-item and user-user similarity simultaneously. Co-clustering methods are also closely related to latent factor models such as nonnegative matrix factorization that simultaneously cluster rows and columns with the use of latent factors.

18.5.5 Latent Factor Models

The clustering methods discussed in the previous section use the aggregate properties of the data to make robust predictions. This can be achieved in a more robust way with latent factor models. This approach can be used either for ratings matrices or for positive preference utility matrices. Latent factor models have increasingly become more popular in recent years. The key idea behind latent factor models is that many dimensionality reduction and matrix factorization methods summarize the correlations across rows and columns in the form of lower dimensional vectors, or *latent factors*. Furthermore, collaborative filtering is essentially a missing data imputation problem, in which these correlations are used to make predictions. Therefore, these latent factors become hidden variables that encode the correlations in the data matrix in a concise way and can be used to make predictions. A robust estimation of the k-dimensional *dominant* latent factors is often possible even from incompletely specified data, when the value of k is much less than d. This is because the more concisely defined latent factors can be estimated accurately with the sparsely specified data matrix, as long as the number of specified entries is large enough.

The n users are represented in terms of n corresponding k-dimensional factors, denoted by the vectors $\overline{U_1} \ldots \overline{U_n}$. The d items are represented by d corresponding k-dimensional

factors, denoted by the vectors $\overline{I_1} \ldots \overline{I_d}$. The value of k represents the reduced dimensionality of the latent representation. Then, the rating r_{ij} for user i and item j is estimated by the vector dot product of the corresponding latent factors:

$$r_{ij} \approx \overline{U_i} \cdot \overline{I_j}. \tag{18.14}$$

If this relationship is true for every entry of the ratings matrix, then it implies that the entire ratings matrix $D = [r_{ij}]_{n \times d}$ can be factorized into two matrices as follows:

$$D \approx F_{user} F_{item}^T. \tag{18.15}$$

Here F_{user} is an $n \times k$ matrix, in which the ith row represent the latent factor $\overline{U_i}$ for user i. Similarly, F_{item} is an $d \times k$ matrix, in which the jth row represents the latent factor $\overline{I_j}$ for item j. How can these factors be determined? The two key methods to use for computing these factors are singular value decomposition, and matrix factorization, which will be discussed in the sections below.

18.5.5.1 Singular Value Decomposition

Singular Value Decomposition (*SVD*) is discussed in detail in Sect. 2.4.3.2 of Chap. 2. The reader is advised to revisit that section before proceeding further. Equation 2.12 of Chap. 2 approximately factorizes the data matrix D into three matrices, and is replicated here:

$$D \approx Q_k \Sigma_k P_k^T. \tag{18.16}$$

Here, Q_k is an $n \times k$ matrix, Σ_k is a $k \times k$ diagonal matrix, and P_k is a $d \times k$ matrix. The main difference from the 2-way factorization format is the diagonal matrix Σ_k. However, this matrix can be included within the user factors. Therefore, one obtains the following factor matrices:

$$F_{user} = Q_k \Sigma_k \tag{18.17}$$

$$F_{item} = P_k. \tag{18.18}$$

The discussion in Chap. 2 shows that the matrix $Q_k \Sigma_k$ defines the reduced and transformed coordinates of data points in *SVD*. Thus, each user has a new set of a k-dimensional coordinates in a new k-dimensional basis system P_k defined by linear combinations of items. Strictly speaking, *SVD* is undefined for incomplete matrices, although heuristic approximations are possible. The bibliographic notes provide pointers to methods that are designed to address this issue. Another disadvantage of *SVD* is its high computational complexity. For nonnegative ratings matrices, *PLSA* may be used, because it provides a probabilistic factorization similar to *SVD*.

18.5.5.2 Matrix Factorization

SVD is a form of matrix factorization. Because there are many different forms of matrix factorization, it is natural to explore whether they can be used for recommendations. The reader is advised to read Sect. 6.8 of Chap. 6 for a review of matrix factorization. Equation 6.30 of that section is replicated here:

$$D \approx U \cdot V^T. \tag{18.19}$$

This factorization is already directly in the form we want. Therefore, the user and item factor matrices are defined as follows:

$$F_{user} = U \tag{18.20}$$

$$F_{item} = V. \tag{18.21}$$

The main difference from the analysis of Sect. 6.8 is in how the optimization objective function is set up for incomplete matrices. Recall that the matrices U and V are determined by optimizing the following objective function:

$$J = ||D - U \cdot V^T||^2. \tag{18.22}$$

Here, $|| \cdot ||$ represents the Frobenius norm. In this case, because the ratings matrix D is only partially specified, the optimization is performed only over the *specified entries*, rather than all the entries. Therefore, the basic form of the optimization problem remains very similar, and it is easy to use any off-the-shelf optimization solver to determine U and V. The bibliographic notes contain pointers to relevant stochastic gradient descent methods. A regularization term $\lambda(||U||^2 + ||V||^2)$ containing the squared Frobenius norms of U and V may be added to J to reduce overfitting. The regularization term is particularly important when the number of specified entries is small. The value of the parameter λ is determined using cross-validation.

This method is more convenient than *SVD* for determining the factorized matrices because the optimization objective can be set up in a seamless way for an incompletely specified matrix no matter how sparse it might be. When the ratings are nonnegative, it is also possible to use nonnegative forms of matrix factorization. As discussed in Sect. 6.8, the nonnegative version of matrix factorization provides a number of interpretability advantages. Other forms of factorization, such as probabilistic matrix factorization and maximum margin matrix factorization, are also used. Most of these variants are different in terms of minor variations in the objective function (e.g., Frobenius norm minimization, or maximum likelihood maximization) and the constraints (e.g., nonnegativity) of the underlying optimization problem. These differences often translate to variants of the same stochastic gradient descent approach.

18.6 Web Usage Mining

The usage of the Web leads to a significant amount of *log* data. There are two primary types of logs that are commonly collected:

1. *Web server logs:* These correspond to the user activity on Web servers. Typically logs are stored in standardized format, known as the *NCSA common log format*, to facilitate ease of use and analysis by different programs. A few variants of this format, such as the *NCSA combined log format*, and *extended log format*, store a few extra fields. Nevertheless, the number of variants of the basic format is relatively small. An example of a Web log entry is as follows:

```
98.206.207.157 - - [31/Jul/2013:18:09:38 -0700] "GET /productA.pdf
HTTP/1.1" 200 328177 "-" "Mozilla/5.0 (Mac OS X) AppleWebKit/536.26
(KHTML, like Gecko) Version/6.0 Mobile/10B329 Safari/8536.25"
"retailer.net"
```