Recommender Systems and their Security Concerns

JUN WANG, QIANG TANG
University of Luxembourg
{jun.wang, qiang.tang}@uni.lu

Abstract

Instead of simply using two-dimensional User × Item features, advanced recommender systems rely on more additional dimensions (e.g. time, location, social network) in order to provide better recommendation services. In the first part of this paper, we will survey a variety of dimension features and show how they are integrated into the recommendation process. When the service providers collect more and more personal information, it brings great privacy concerns to the public. On another side, the service providers could also suffer from attacks launched by malicious users who want to bias the recommendations. In the second part of this paper, we will survey attacks from and against recommender service providers, and existing solutions.

1. Introduction

Recommender systems have attracted researchers’ attention since the early of 1990s, e.g. [112, 160, 173]. They have been applied to a variety of fields (e.g. news, online multimedia, e-commerce, tourism and social network) to handle information overload and provide personalized services. In the early days, researchers focused on two-dimensional User × Item features. At that time, it’s not as easy as nowadays to collect additional features, due to the limited computation resources and lack of knowledge. Later, researchers found that additional information can significantly enhance recommender systems [11]. For example, demographic information can be used to mitigate cold start problem, temporal information can be applied to capture users’ shifting interests, and location and social network information can be used not only to improve recommender systems but also to create new recommender systems [3, 74, 204]. Many researchers have investigated how to integrate multiple features into the recommendation algorithms to provide better service, e.g. [11, 108, 122, 120].

On the flip side, recommender systems bring great privacy concerns to the public, as surveyed in [14]. Some researchers have tried to seek solutions which approximate the outputs without losing accuracy and effectiveness (or, utility) but protecting individual’s privacy. In reality, this is dilemma, there is always a tradeoff between privacy and utility. The differential privacy paradigm [48] provides a rigorous framework to quantify the privacy loss. If someone expects to leak less personal information, he has to add more noise to the output or dataset. As a result, the accuracy will be degraded. Some other researchers proposed cryptographic solutions which do not affect utility. But, they are often computationally expensive. As it mentioned in [199], it has a six orders of magnitude difference between the cryptographic operations and plaintext arithmetic operations. Another different concern for recommendation services is robustness. The service providers are facing attacks from malicious users who try to bias the recommendations [111]. An example is that, someone wants his book to be recommended in a top rank, so he forges a group of users and inserts them to the recommender system’s database. These forged users will give his books high ratings and assign his competitors’ books low
scores. This will finally degrade recommender systems’ reputation and users will distrust the recommender system.

A number of studies on all aspects of recommender systems have been published. [124] surveys the content based algorithms. [176] introduces a variety of collaborative filtering techniques. [17] gives an overview of hybrid recommender systems. [91] investigate the robustness of recommender systems. [14] [185] focus on the privacy concerns. [11] presents the possible extensions of recommender system, and shows the importance of multi-dimensionality of recommendations and multi-criteria ratings.

In this paper, we try to fill in the gap between the literature work and most recent advances in recommender systems. Firstly, we survey the recent recommender systems which utilize multi-dimensional features to improve their performances. Secondly, we provide a general survey on the robustness and privacy issues facing recommender systems, and present some representative solutions. As a result, we wish to give the researchers a clear idea on the state of the art of this field.

In the rest of the paper, we give an overview of the state-of-the-art for recommender systems in Section 2. We present the main features (dimensionalities) involved in recommender systems in Section 3. We describe the robustness concerns in Section 4 and privacy concerns in Section 5. We conclude the paper in Section 6.

2. Review of Recommender Systems

We reserve the following characters in our paper to denote specific objects. \( U \) denotes a set containing \( n \) users, and \( u \) is a user in \( U \), namely \( u \in U \). \( I \) denotes a set containing \( m \) items, and \( i \) is an item in \( I \), namely \( i \in I \). \( R = U \times I \) is the User \( \times \) Item matrix that contains all the ratings that the users have assigned to the items. \( \bar{R}_x \) indicates the rating vector of user (item) \( x \). \( R(u, v) \) denotes all the items are rated by both user \( u \) and \( v \). \( R(i, j) \) means all the users rated both item \( i \) and \( j \). \( \hat{r}_{ui} \) denotes the predicted value that user \( u \) will assign to item \( i \). \( \bar{r}_u \) is the mean rating value of user \( u \). \( \bar{r}_i \) is the mean rating value of item \( i \). \( \kappa \) denotes a set containing all the available \((u,i)\) pairs in a dataset, it means \( r_{ui} \in \kappa \) has been provided by user \( u \).

2.1. Definition of Recommender Systems

Simply, recommender system is a tool that can help users to extract high valued and personalized information among massive data sources. In mid-1990s, recommender system became an independent field, and researchers started focusing on recommendation with explicit feedback [11]. In a common case, the recommendation task is to predict users’ preference based on history information. Intuitively, given a bunch of users and their historical behavior, the recommender system estimates a user’s preferences on items which have not appeared in her record. A typical way of using the historical information is to transfer it to a User \( \times \) Item matrix, as shown in Table 1 where ratings are specified on a range from 1 to 5. The symbol \( ? \) says that the item has not been rated by the user (alternatively, people often use 0 to indicate items are not rated).

<table>
<thead>
<tr>
<th>Item</th>
<th>Item2</th>
<th>Item3</th>
<th>Item4</th>
</tr>
</thead>
<tbody>
<tr>
<td>User1</td>
<td>5</td>
<td>3</td>
<td>?</td>
</tr>
<tr>
<td>User2</td>
<td>?</td>
<td>4</td>
<td>?</td>
</tr>
<tr>
<td>User3</td>
<td>3</td>
<td>?</td>
<td>5</td>
</tr>
<tr>
<td>User4</td>
<td>?</td>
<td>4</td>
<td>?</td>
</tr>
<tr>
<td>User5</td>
<td>1</td>
<td>?</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 1: A typical user-item matrix

Formally, an abstraction of recommender systems, showing in Figure 1 can be described as following: Let \( U \) be a set contains \( n \) users, \( I \) be a set has \( m \) items which are collected from \( U \)’s history, and \( R = U \times I \). Define the recommender system as:

\[
f : R^{n \times m \times d} \rightarrow \hat{R}^{n \times m}
\]

\( d \) is number of dimension features. The ratings are the infrastructure of the system. \( d = 1 \) indicates this is a two-dimensional User \( \times \) Item space. If additional features, like time, social interaction and so on, are involved, \( d - 1 \) the number of additional features. \( f \) is the recommendation algorithm, and it decides which dimensionalities will be involved in the computation and how to use them. \( \hat{R} \), output of function \( f \), is the prediction matrix.
Regarding the outputs of a recommender system, there are usually two possibilities.

- Single prediction, it outputs a precise prediction score that a user will give to an unrated item.
- Top-N prediction, it outputs $N$ unrated items that a user would prefer the most.

A general classification of existing recommendation methods is as follows [7], also shown in Figure 2.

- Content based: it calculates the items’ similarity based on their contents, such as texts in books, acoustical signal of music. It then recommends the items which are most similar to those the users consumed before.

- Collaborative filtering: it relies on the correlations between users or items. For example, it suggests those items to a user that the people similar to this user also prefer.

- Hybrid approach: it combines different kinds of recommender systems, like composing content based and collaborative filtering methods, or different kinds of collaborative filtering methods together.

2.2. Content Based Recommender

Content based methods employ the attributes of unrated items to match a user’s profile which is constructed based on the user’s history behavior. For example, if a user has bought a book $A$, the recommender will find some books which are similar to book $A$, and recommend these books to him. Formally, content based methods can be described as:

$$ f : (\text{User-Profile}, \text{Content-Profile}) \rightarrow \tau $$

where $\tau$ is a metric to measure how much a user like an item.

Existing content based methods can be classified into two categories: traditional heuristics methods which are mostly based on information retrieval approach [6] [165], and model based methods such as Bayesian classifiers, clustering, topic model and so on.

Text information of items is easy to find in the real word, therefore many content based methods are focusing on text-based application, like the content of books, web sites, lyrics of musics and so on. The units (components) are often called keywords.

Term Frequency/Inverse Document Frequency (TF-IDF) [166] [165] is one the most popular measures for weighting keywords in information retrieval. Intuitively, Term Frequency ($TF$) is the frequency of the keyword exists in a document. Let $f_{ij}$ denote the number keyword $k_i$ appears in document $d_j$. Formally, the normalized frequency is defined as:

$$ TF_{ij} = \frac{f_{ij}}{\max_z f_{zj}} $$

$max_z f_{zj}$ is the max frequency over any keywords $k_z$ in document $d_j$. It’s not enough to only use $TF$ as the measurement of keywords,
because if some keywords exist in most of the documents then they may not be useful to distinguish a relevant document and a non-relevant document. Combined with Inverse Document Frequency (IDF), the importance of keywords can be measured much more properly. IDF is defined as:

$$IDF_i = \log \frac{N}{n_i}$$

$N$ is the total number of the documents. $n_i$ is the number of documents where the term $i$ appears. The TF-IDF weight for keyword $k_i$ in document $d_j$ is defined as:

$$w_{ij} = TF_{ij} \times IDF_i$$

Besides the statistical information gained by basic TF-IDF for each term, many researchers also employ semantic analysis to enrich these terms, like providing culture and linguistic background knowledge to enhance their recommender systems’ performance [51] [39] [57]. Overall, the profile of document (item) $i$ is denoted by a vector as:

$$Content-Profile(i) = [w_1, w_2, \ldots, w_n]$$

In order to recommend similar items to users, recommender system has to construct users profile by the items that users have rated, ordered or read. Formally, define User-Profile($u$) be the profile of user $u$:

$$User-Profile(u) = [w_{1u}, w_{2u}, \ldots, w_{nu}]$$

There are many metrics that can be employed to measure the similarity between two vectors. Cosine Similarity is one of the most widely used metrics:

$$\text{cosine}(W_u, W_i) = \frac{W_u \cdot W_i}{\|W_u\|_2 \times \|W_i\|_2}$$

where $W_u$ denotes User-Profile($u$), and $W_i$ indicates Content-Profile($i$). After all, the items with relative higher similarity will be recommended to a user.

Instead of employing information retrieval methods, many researchers studied using machine learning methods for prediction. Naive Bayesian classifier [47] is a widely used classier to estimate the probability that a document (item), which contains a set of terms \( \{k_1, k_2, \ldots, k_n\} \), belongs to a specific class $c$:

$$p(c|k_1, k_2, \ldots, k_n)$$

With the assumption that the terms are independent, the probability can be estimate by:

$$p(c) \prod_i^n p(k_i|c)$$

$p(c)$ and $p(k_i|c)$ can be calculated by the training data [152].

To sum up, content based methods require analysis of the features of the underlying items. For some items, such as books and news, it is easy to get the content features. But for others, like videos and music, more efforts are required. The process is often complicated and expensive in computation. To alleviate this problem, other methods can be used, like Singular Value Decomposition (SVD) [66] and Principal Component Analysis (PCA) [98].

2.3. Collaborative Filtering

Collaborative filtering is one of the most successful methods [176] to build recommender systems. It mines the correlations among users or items to predict a user’s preferences on new items. Classically, this method can be further categorized into memory based approach and model based approach.

Memory based. Memory based approach recommends items based on the entire collection of items which have been rated by users previously. Intuitively, given a user $u$, the prediction of his unrated items is computed based on the users who are similar to him or the items similar to what he has rated before. There are a number of metrics introduced to measure similarity between users or items [176] [172]. Next, we take Pearson correlation coefficient and Cosine similarity as examples to review user similarity. The definitions can be easily modified to calculate the similarities between items.

- Pearson correlation coefficient is the covariance of the two vectors divided by the product of their standard deviations. It gives a value in the range $[-1, 1]$, where 1 is total positive correlation, 0 denotes...
no correlation, and -1 indicates total negative correlation.
\[
\text{sim}(u, v) = \frac{\sum_{i \in I_{uv}} (r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum_{i \in I_{uv}} (r_{ui} - \bar{r}_u)^2} \sqrt{\sum_{i \in I_{uv}} (r_{vi} - \bar{r}_v)^2}}
\]

\( I_{uv} \) denotes the set of items are rated both user \( u \) and \( v \), and \( r_{ui} \) and \( r_{vi} \) denote the scores that user \( u \) and \( v \) gave to item \( i \) respectively.

- Cosine similarity is a measure of similarity between two vectors of an inner product space that measures the cosine of the angle between them. The output is in the range \([0, 1]\).

\[
\text{sim}(u, v) = \frac{\sum_{i \in I_{uv}} r_{ui}r_{vi}}{\sqrt{\sum_{i \in I_{uv}} r_{ui}^2} \sqrt{\sum_{i \in I_{uv}} r_{vi}^2}}
\]

Among all memory based methods, neighborhood based method is a representative [41]. This method can be either user based [105, 27] or item based [168, 118]. Mathematically, they are very similar so that we will introduce them in a uniform manner. In the following, we refer to the definitions from [115, 105, 168]:

\[
\hat{r}_{xy} = \frac{\sum_{x' \in X} \omega_{x'x} \cdot r_{x'y}}{\sum_{x' \in X} \omega_{x'x}}
\]

\[
\hat{r}_{xy} = r_x + \frac{\sum_{x' \in X} \omega_{x'x} \cdot (r_{x'y} - \bar{r}_{x'})}{\sum_{x' \in X} \omega_{x'x}}
\]

Let \( \bar{r}_x \) be the mean value of \( \{r_{x,y}, x \in X\} \). \( \omega_{x'x} \) is the similarity between \( x \) and \( x' \). If it is user based, \( r_{x,y} \) represents the score that user \( x \) gave item \( y \), and \( X \) is the neighbor set of users who have rated item \( y \). If it is item based, \( r_{x,y} \) represents the score that user \( y \) gave to item \( x \), \( X \) is the neighbor set of items which have been rated by user \( y \).

Model based. In contrast to memory based approach, model based approach learns the history data and models it [31, 131, 109, 139]. In recent years, model based approach has been intensively studied and become very popular. Matrix Factorization [113, 109] is demonstrated to be very efficient and stable, and it is one of the most popular methods in the category of model based approach. In practice, it helps Koren [109] to win the Netflix Prize\(^1\) in 2009.

The intuition behind using matrix factorization is that users rate a movie will depend on some latent features, such as, whether they like the actors/actresses or the genre of the movie. It is reasonable to imagine that if the recommender systems can discover more such factors then it will generate more accuracy recommendation. Naturally, the number of factors should be less than items number and users number. Formally, suppose that there are two latent factor matrices \( P (|U| \times K) \) and \( Q (|I| \times K) \) that their production can approximate \( R \) with a tolerable error:

\[
R \approx P \cdot Q^T = \hat{R}
\]

Each row of \( P \) represents the strength of association between a user and the latent factors. And each row of \( Q \) represents the correlation between an item and the latent factors. The prediction of the rating user \( u \) assigns to item \( i \) can be estimated by:

\[
\hat{r}_{ui} = P_u Q_i^T = \sum_{k=1}^{K} p_{uk} q_{ki}
\]

A cost function, which is used to measure the error between real rating and estimated rating, is defined as:

\[
e^2_{ui} = (r_{ui} - P_u Q_i^T)^2 + \beta \cdot \sum_{k=1}^{K} (||P_u||^2 + ||Q_i||^2)
\]

Parameter \( \beta \) is used to control the magnitudes of the user latent factors and item latent factors. Now, the task of obtaining \( P \) and \( Q \) becomes to minimize the cost function of the all \((u,i)\) pairs \( \kappa \):

\[
\min \sum_{(u,i) \in \kappa} e^2_{ui}
\]

Stochastic gradient descent\(^2\) (SGD) is a very popular method to solve the above cost function. It’s an iterative algorithm, the update rules

\(^1\)http://www.netflixprize.com/
\(^2\)http://sifter.org/ simon/journal/20061211.html
for $P_u$ and $Q_i$ are:

$$P_u = P_u + \alpha \frac{\partial}{\partial P_u} e_{ui}^2 = P_u + \alpha (2e_{ui}Q_i - \beta P_u)$$

$$Q_i = Q_i + \alpha \frac{\partial}{\partial Q_i} e_{ui}^2 = Q_i + \alpha (2e_{ui}P_u - \beta Q_i)$$

$\alpha$ determines the rate of approaching the minimum. $\alpha$ and $\beta$ are empirical values.

In the rest of this survey, when mentioning matrix factorization, we will use predication function or cost function directly, which can be easily obtained from each other under the methodology we described above.

**Improvement.** Based on the basic matrix factorization model, researchers have also introduced a variety of other factors, like neighborhood, global average, item bias, user bias, time and so on, to construct rating prediction function to achieve better performance [109, 106]. [139] extends a probabilistic matrix factorization model which assumes that $r_{ui}$ is subject to a normalized distribution.

**Challenges.** Collaborative filtering, as one of the most successful approaches to build recommender systems, also has many challenges. This approach may suffer from cold start problem [170], which means it can not provide accurate recommendation service to new users or items. This problem can be mitigated by hybrid approach based systems and additional information, like demography and social network. Scalability is another issue. In reality, the number of users and items can be very large, such as millions of users and items. What makes the situation more serious is that some applications require real-time recommendation services, like news recommendation [35]. Data Sparsity [85] is yet another issue. In most cases, although the number of users and items are large, each user only rates a small portion of the available items. It’s difficult to find similarities among different users or items.

### 2.4. Hybrid Approach

Hybrid recommender system [17] combines two or more recommendation methods to mitigate certain limitations of individual method and improve accuracy. Two main approaches exist to implement hybrid recommender systems:

- Using model based composition methods, like linear regression [11, 12], neural network [107] and so on to composite the predictions generated by a variety of recommendation methods.

- Constructing a comprehensive model that integrates different features and methods together [106, 120, 170, 34, 68].

**Model based composition.** In the model based composition, suppose that the recommender system employs $K$ models. [11] suggests that using a simple linear model to composite multiple predictions $X$:

$$f(X) = X\beta$$

$\beta \in \mathbb{R}^K$ is the regression coefficient. It can be obtained by minimizing the prediction error:

$$||X\beta - b||^2_x + \lambda ||\beta||^2_2$$

The final rating is:

$$\hat{r}_{ui} = \sum_{k=1}^{K} \beta_k \cdot \hat{r}_{ui}^k$$

[12] introduces a more sophisticated linear regression based method to better model certain users or items. [17] describes a cascade model that one recommender’s output is another recommender’s input. Other methods have also been developed to composite predictions, like neural network, polynomial regression, gradient boosted decision tree and so on [107, 107, 154].

**Single comprehensive model.** In contrast to model based composition, a comprehensive model tries to integrate different kinds of factors together to construct the recommender system. It may merge content based method and collaborative filtering, or different kinds of collaborative filtering methods to generate recommendation.

[106] described a way to merge neighborhood based method with latent factor model smoothly. Simply, they employ a neighborhood method as following:

$$\hat{r}_{ui} = b_{ui} + \sum_{j \in R_u \setminus i} (r_{ui} - b_{uj})w_{ij}$$
Express latent factor model as:

$$\hat{r}_{ui} = b_{ui} + P_u Q_i^T$$

Then construct the integrated model, as:

$$\hat{r}_{ui} = b_{ui} + P_u Q_i^T + \sum_{j \in R_u \setminus i} (r_{ui} - b_{uj}) w_{ij}$$

$b_{ui}$ is a baseline estimate for an unknown rating: $b_{ui} = \mu + b_u + b_i$, where $\mu$ is the overall average rating, $b_u$ and $b_i$ indicate the observed deviations of user $u$ and item $i$, respectively. The parameters of the integrated model can be estimated by minimizing the associated regularized squared error function (cost function) through gradient descent method as we described before.

Researchers also incorporated content based method with collaborative filtering method [120, 186]. Take [120] as instance, it first classifies the items (news) category, then employs Bayesian classifier to determine a user’s interest to the specific news. Finally, it is incorporated with the collaborative filtering output [36]: $Rec(user, item) = CR(user, item) \times CF(user, item)$. CR denotes content based recommender and CF is collaborative filtering.

Summary. Hybrid recommender systems can give more accurate prediction than systems with single model, like a pure content based method or a pure collaborative filtering approach. It’s helpful to alleviate cold start problem, to mitigate attack from malicious users. It can improve the effectiveness of scalability [36]. However, on the other hand, it requires more resources including data sources and computations. Some papers, e.g. [153, 138], empirically compare the performance of hybrid with pure content based or collaborative methods.

2.5. Metrics

The quality of recommender systems are verified by the pre-defined metrics. They are well studied in [172, 76]. In this paper, we only introduce the widely used prediction accuracy metrics.

Mean Absolute Error (MAE)

$$MAE = \frac{\sum_{(i,j) \in \kappa} |\hat{r}_{ij} - r_{ij}|}{N}$$

where $N$ is the total number of ratings over all the users in test set. $\hat{r}_{ij}$ is the estimation for user $i$ over item $j$. $r_{ij}$ is the real rating. MAE measures the absolute error between the estimated and the real ratings.

Root Mean Squared Error (RMSE)

$$RMSE = \sqrt{\frac{\sum_{(i,j) \in \kappa} (\hat{r}_{ij} - r_{ij})^2}{N}}$$

RMSE amplifies the absolute error between the predicted and the true ratings. So it’s a good choice when the errors can have great impact on users’ decision.

Precision and Recall

Precision is the ratio of retrieved instances that are relevant. It is defined as:

$$Precision = \frac{T_P}{T_P + F_P}$$

where $T_P$ is the number of true positives, and $F_P$ is the number of false positives. Recall is the portion of relevant instances that are retrieved, the definition is:

$$Recall = \frac{T_P}{T_P + F_N}$$

$F_N$ is the number of false negatives. F1-score is defined as the harmonic mean of precision and recall, which is employed to consider the influence of both Precision and Recall:

$$F_1 = \frac{2 \times Precision \cdot Recall}{Precision + Recall}$$

Receiver operating Characteristic (ROC) curves

ROC curve illustrates the performance of a binary classifier system in a two-dimensional space. The curve presents the true positive rate (TPR) against the false positive rate (FPR) in various threshold settings. It is widely used in signal processing. [76] employed it to distinguish good predictions from bad predications. The Area under the ROC Curve (AUC) is used to measure a system’s capability of distinguishing good predictions from bad predications. Bigger value of AUC indicates better performance of the system.
3. Dimensions of Recommender System

Most of the first generation recommender systems focused on two-dimensional User × Item matrix. However, researchers soon realized that considering more additional information may bring significant benefit, like mitigating cold-start problem, increasing accuracy, enhancing stability. We survey these main dimensions for some popular application areas of recommender systems and summarize them in Table 2. In the following, we consider cost of money, tags, emotion, demography as Context.

3.1. Explicit Feedback and Implicit Feedback

Users’ explicit feedback on items is the most direct approach to construct User × Item matrix. For example, users rate a movie on Netflix with a score in the range [1,5], and use 0 to denote the fact that a movie is not rated. But in reality, only a small portion of the items can receive explicit feedback. So collecting implicit feedback, such as news click history, music play count, visiting history of a place, video watch history and so on, is crucial to construct User × Item matrix to build recommender systems. Generally, researchers always try to quantify implicit feedback into a numeric matrix. Thus, almost all the collaborative filtering methods can be employed to build recommender systems based on it. For instance, treats a click event as a vote to news. It creates binary matrix to present users’ preference on news. assumes that users listen to songs more often if they like them, so it employs play count as rating which a user assigns to a song.

Next, we will introduce some representative algorithms. A very basic neighborhood based method is described as follows:

\[
\hat{r}_{ui} = \frac{\sum_{j \in R_u \cap R_i} w_{ij} r_{ij}}{\sum_{j \in R_u \cap R_i} w_{ij}}
\]

presents a model called Slope one. It emphasizes the influence of differences between items and users’ rating habits. For instance, some users prefer to give higher score to items, but others would like to assign lower score. A mathematical definition is:

\[
\hat{r}_{ui} = \frac{\sum_{j \in R(u) \cap R(i)} (\delta_{ij} + u_i) w_{ij}}{\sum_{j \in R(u) \cap R(i)} w_{ij}}
\]

where \(\delta_{ij}\) is defined as:

\[
\delta_{ij} = \sum_{u \in R(i,j)} (u_i - u_j) / |R(i,j)|
\]

\(R(i,j)\) denotes a set of users who rated item \(i\) and \(j\).

Other usage of ratings, such as average rating of a user/item, rating number of a user/item and so on, are further discussed in 10 [107]. These features are uniformly called global effects. Quite often, a recommender system combines multiple global effects in sequence. When it learns a global effect, it takes as input the prediction residual generated by all the previous global effects (only the first global effect uses original ratings).

As we presented in Section 2.3, matrix factorization is one of the most successful methods of recommendation. It has many variants, all of them can be solved following the same methodology we introduced in Section 2.3. Here, we describe a very well-known variant proposed in 159. It incorporates user bias and item bias to matrix factorization model. A well-known mathematical definition is from 109:

\[
\hat{r}_{ui} = \mu + b_u + b_i + P_u Q_i^T
\]

where \(\mu\) denotes overall average, \(b_u\) and \(b_i\) indicate the observed deviations of user \(u\) and item \(i\) from the average, respectively. The probabilistic matrix factorization 139 assumes that the ratings follow Gaussian distribution, and it further assumes that different users (items) are independent to each other. A formal description is as follows:

\[
p(R|P, Q, \delta^2) = \prod_{u=1}^{N} \prod_{i=1}^{M} \prod_{(R_{ui} \neq 0)} [N(R_{ui}|P_u Q_i^T, \delta^2)] I_{ui}
\]

where \(N(R_{ui}|P_u Q_i^T, \delta^2)\), generally denoted as \(N(x|\mu, \delta^2)\), is the probability density function of the Gaussian distribution with mean \(\mu\) and variance \(\delta^2\). \(P_u\) and \(Q_i\) represent user and item latent features respectively. \(I_{ui}\) indicates...
whether item \( i \) has been rated by user \( u \). To obtain the prediction, this model’s cost function is estimated by maximizing the posterior distribution over the user and item features [139, 163].

Association rule mining [4] is an important technique in recommendation to discover interesting relations between items. [37] uses this (called co-view) to measure how often a pair of items are visited in a session (the same user visits those items in a certain period). [164] describes a class of two-layer undirected graphical models that generalize Restricted Boltzmann Machines (RBM) [175, 80] to model User × Item matrix. It uses Contrastive Divergence [79] to learn a model for prediction. For large scale recommender systems, clustering approach is often adopted to avoid unnecessary computation, so that only highly related items or users are involved. [36] employs Local Sensitive Hashing (LSH) [92] and probabilistic latent semantic model [82] to cluster users into different groups (topics). Note that a user can belong to multiple groups. Parallel based and distributed based methods are also sharp tools to improve the effectiveness of large scale datasets, and are extensively studied in [206, 60, 159, 207].

### Table 2: Features used in different areas

<table>
<thead>
<tr>
<th>Explicit feedback</th>
<th>News</th>
<th>Tourism</th>
<th>Movies &amp; Video</th>
<th>Music</th>
<th>Books</th>
<th>Social network</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implicit feedback</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Time</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Content</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>Cost</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Location</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>Social interaction</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>Demography</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>Tags</td>
<td></td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>emotion</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td></td>
</tr>
</tbody>
</table>

3.2. Content

Content, as an independent feature, can be used to construct pure content based recommender systems or combine with collaborative filtering methods to build hybrid recommender systems. As the two folds are already discussed in Section 2.3 and Section 2.4 in this section we present the representative application areas indicated in Table 2.

[138] presents a way to combine content with User × Item rating matrix to build collaborative filtering recommender systems. It first creates a pseudo user-ratings vector for each user in the dataset, and then form the new User × Item matrix as:

\[
\begin{align*}
    r'_{ui} &= \begin{cases} 
        r_{ui} & : \text{if user } u \text{ has rated item } i \\
        c_{ui} & : \text{otherwise}
    \end{cases}
\end{align*}
\]

where \( c_{ui} \) denotes the rating predicted by pure content based recommender system. Then any collaborative filtering methods can be applied to the resulted User × Item matrix.

[142] constructs user profile based on the books that the user has ever rated. It employs Bayesian method to measure the strength of words (from content, title, authors) which are projected to a specific user. Then, the system uses the strength of words to influence the final recommendation. [120] uses news content to classify a user’s click history. Latent Dirichlet Allocation (LDA) [16] is a very efficient and important method to detect content topics. In [117], researchers first cluster news, then employ LDA to detect latent topics, and use a vector to present the topic distribution of a news in the same group. [119] uses LDA to cluster Places of Interest by their textual description.

For multimedia recommendation, like music, video and movie, the content features are very important. Many methods have been explored to extract this kind of features [69, 114, 89, 42, 78]. [183] uses some conventional approaches of music information re-
which correspond to "long term" and "short term" interests, are investigated to improve news recommendation. [36] divides the time serial into several bins, and predicts personal click distribution (the probability of a news belongs to a specific category $c$) for each bin $t$, $p^t(\text{category}=c)$. Then, it combines the results of all the bins to obtain genuine interests. To obtain news trend, [36] uses the click distribution of the history data in a very short time slot, like 1 hour or 10 minutes.

In tourism, some places of interest may be more attractive in a certain season, and people may lean to hang out for tour in some specific periods. Temporal information can be extracted from visit history [122]. Weight Average Entropy (WAE) [93] can be applied to find the best splitting of the tour time serial, as [122]:

$$WAE(S^p, i) = \frac{|S^p_1(i)|}{|S^p|} \cdot \text{Ent}(S^p_1(i)) + \frac{|S^p_2(i)|}{|S^p|} \cdot \text{Ent}(S^p_2(i))$$

It treats the entire year as a big season at the initialization phase and then partitions it into several seasons in a recursive binary way. $\text{Ent}(S^p)$ is the entropy information of season $S^p$, the definition is: $\text{Ent}(S^p) = -\sum_{i=1}^{|S^p|} p_i \log(p_i)$. $|S^p|$ is the number of travel packages (items) in season $S^p$. $p_i$ is the proportion of travel package $P_i$ in season $S^p$. $S^p_1$ and $S^p_2$ are two sub-seasons of season $S^p$ when being split at $i$-th month. By seeking a maximum information of $\Delta \text{Ent}(i)$ which is defined as: $\text{Ent}(S^p) - WAE(S^p, i)$, the system can find the best month splitting. Besides finding the best time splitting described as above, the duration of time is also considered into recommender system [184] and it is treated as a kind of cost.

3.4. Location

With the development of position localization techniques, people can share their location with others easily. Researchers have leveraged on location information to enhance a recommender system’s performance and to provide new location recommend services. Two main hypothesis are important to location based recommender systems.

- Tobler’s proximity law: everything is related to everything else, but near things are more related than distant things [180].
- Power law distribution: it indicates the probability of a user to check-in a place decays as the power law of the distance between them.

Typically, location features can be classified into three categories.

- Location history, such as users’ location check-in history and online rating history of locations [32, 83, 197].
- User Trajectories, such as the visiting sequence, travel path, GPS trajectory and so on [116, 25, 203, 205].
- Location tag, such as location information in social media [73, 71, 198].

In tourism recommendation, geographical information plays a very important role. [197] uses check-in (visit) history to construct a user item matrix. 1 stands for a user visited a place, and 0 indicates a place is not visited by the user. Then, any collaborative filtering method can be applied to the matrix. It analyzes the check-in probability to the distance between two places visited by the same user, and employs Bayesian method to predict the probability of visiting a new place based on the user’s visit history. [119] uses check-in count data to form a user item matrix, it integrates spatial influence and user mobility to predict users’ check-in decision. Compare to location history, user trajectories, such as travel path and GPS trajectory, contain more information. In the meanwhile, the number of locations in continuous trajectory can be numerous. Therefore, how to identify highly relevant location information is an important topic [116]. [205] learns the location correlations from a large number of user-generated GPS trajectories to provide location recommendation. The correlation between two locations is calculated by integrating the travel experiences of the users who have visited them in a trip in a weighted manner. Instead of using distance to decay the weight of the location, [205] decreases the location’s weight as the interval between the two locations’ index. It describes a Tree-Based Hierarchical Graph (TBHG) structure to model users’ GPS trajectory data, and a HIT-based influence model [102] to measure the representation of locations and the travel experience of users. In practice, location features can be hidden or attached in media content. [125, 177] extract location information from geo-tagged photos to recommend travel route.

In music recommendation, researchers have leveraged on the location information to enhance the system’s performance. [21] extracts the geographical information, like city or country name, from lyrics. [200] uses geospatial positions to recommend music that matches a specific environment. [169] computes a centroid of each user’s geospatial listening distribution, then recommends artists who are close to the user.

As indicated in [8], location correlation is a very strong social property. Intuitively, users with similar location history are more likely to have similar preferences and become friends. [40] reveals that users’ social connections are highly related to their location. [97] extracts geographical information from tweets, and uses the distance of between two tweets to measure their relevance. [33] uses weighted Voronoi diagram to value the users’ location similarity, where a user’s Voronoi diagram is weighted by the user’s dwell time. According the diagram, they construct affinity matrix to recommend friends to user. [90] adopts a probabilistic suffix tree as a trajectory profile to reflects a user’s moving behavior. They discover user communities based on the similarity measurement derived from trajectory profiles of users. [192, 193] hierarchically cluster users into communities based on the similarity constructed from users’ location history.

3.5. Social Interaction
Social networks have gained remarkable attention since mid-1990s [101, 100, 189]. A fundamental observation is homophyly [132] which indicates people with similar interest tend to connect to each other and people with similar interests are likely to be friends. On one hand, researchers have tried to incorporate social information into collaborative filtering methods to improve their performances. On the other hand, social recommendation services have been proposed to recommend friends and discover communities.
People often quantify a social network by transferring it into trust network as Figure 3. Some works assume that users in a social network explicitly state their trust on their friends. Consider the reality that users only state the trust values on a small portion of their friends. Just as predicting the rating value a user will give to an item, neighborhood based methods [129, 64, 63] and matrix factorization [127] are two widely used approaches to estimate users’ trust values on their friends. Some researchers also use implicit correlations, such as followship in Twitter and Like action in Facebook, to construct trust matrix [188]. In the extreme situation, users in a dataset will be isolated. For each user, only the associated events with timestamps are recorded. Some people believe that a latent social network is hidden in these isolated data. [67] employs probabilistic model to infer the network on the diffusion data under an assumption that the weight of the edges in the network are homogeneous. [143] removes this assumption, and presents a maximum likelihood approach based on convex programming to infer the latent weighted social networks from diffusion data. Next, we discuss some representative works in making use of social information for recommender systems.

[127] integrates trust between users in social network into recommender based on matrix factorization method. Suppose that the model has User × Item rating matrix R and User × User trust matrix S, where $S_{uv} \in (0, 1]$ and 0 indicates user $u$ does not trust $v$ at all, 1 means user $u$ completely trust $v$. Both of to matrices are sparse. Their prediction equation is:

$$\hat{R}_{ui} = P_uQ_i^T, \quad \hat{S}_{uv} = P_uZ_v^T$$

where $P, Q, Z$ are user, item and trust latent factors, respectively. The cost function is defined as:

$$\sum_{(u,v) \in \mathcal{X}} (r_{ui} - \hat{r}_{ui})^2 + \sum_{u,v} (s_{uv} - \hat{s}_{uv})^2$$

$$+ \beta(||P||^2 + ||Q||^2 + ||Z||^2)$$

[126] proposes a linear combination of basic matrix factorization approach. It uses the observed trust matrix $S$ directly:

$$\hat{r}_{ui} = \alpha P_uQ_i^T + (1 - \alpha) \sum_{v \in F_u} P_vQ_i^T$$

where $F_u$ is the set of user $u$’s direct friends. $S_{uv}P_vQ_i^T$ in the equation captures social influence. And $\alpha \in [0, 1]$ is used to balance strength of rating information and the social influence. [95] further incorporates trust propagation into the matrix factorization, and the cost function is defined as:

$$\sum_{(u,v) \in \mathcal{X}} (r_{ui} - \hat{r}_{ui})^2 + \gamma \sum_{u \in U} ||Q_u - \sum_{v \in F_u} S_{uv}P_v||^2$$

$$+ \beta(||P||^2 + ||Q||^2)$$

One approach to integrate social influence into neighborhood based method is using trust values between users to substitute similarity values between users [129, 64]. Random walk can also be employed to predict user $u$’s rating on item $i$. For example, [24] chooses $u$ as the source node, and performs several random walks on $u$’s trust network, and the aggregation of all ratings returned by those random walks are considered as the predicted value. In each walk, if user $v$ has rated a target item $i$, then stop walking and return $r_{vi}$. If $v$ has not rated item $i$, it has a probability, which is related to the maximum similarity between target item $i$ and the items $v$ has rated, to continue the walk. If the walk stops, randomly select item $j$ similar to target item $i$ and return $r_{vj}$. [194] finds K nearest neighbors, called CF-Neighbors, from latent user factors obtained by matrix factorization and gets K closet neighbors who are not in CF-Neighbors from the trust network.
Then users in the combined neighborhood vote for their relevant items to recommend Top-N items to target users.

New recommendation services, such as finding friends and discovering communities, is another important branch. The methods used to estimate users’ trust social network can be directly employed to recommend friends. For example, users who trust each other with a high probability are likely to be friends. An observation of social network is that a user’s social life is multifaceted. For instance, a user may trust some of her friends when she wants to buy a computer, and she may trust some other friends when deciding to buy some clothes. It’s important to infer the communities that a user belongs to. \[195\] introduces a set of methods to infer category-specific circles (communities) of friends based on the following definition: if a user \( v \) is in the inferred circle \( c \) of user \( u \), then two minimum request should be satisfied: (1) \( S_{uv} > 0 \) in the social network; (2) \( N_u^c > 0 \) and \( N_v^c > 0 \), where \( N_u^c \) denotes the number of items in category \( c \) that user \( u \) rated. \[43\] models social network as a Markov random fields, which incorporates users’ social influence to predict potential customers of markets. \[26\] designs a system to help users find known, offline contacts and discover new friends on social networking sites.

### 3.6. Context

Recently, context-aware recommender systems have been studied extensively \[2\]. The main purpose is to leverage additional information to enhance recommender systems. The multifaceted nature of context information makes it hard to have a unified definition. A number of different definitions of context exist in the literature \[9,44\]. In this paper, we refer context to the properties belonging to an item, but not the item itself. For example, the content of a book is not context, but the information like its authors, publisher, publishing time and so on are context. And the acoustic signal and the lyrics of a song is not context, the artists or tags attached to the song are context. Based on this, we present the representative application areas of context, indicated in Table\[2\].

### Cost

In tourism recommendation, \[59\] discovers that, besides time, financial cost is also a crucial feature to affect users’ decision. It proposes two cost aware latent factor models to recommend travel packages by incorporating financial cost. This first one is probabilistic matrix factorization model, see Section\[3.1\] it defines the mean \( \mu \) of the probability density function, \( \mathcal{N}(x; \mu, \sigma^2) \), of the Gaussian distribution as:

\[
\mu = f(P_u, Q_i, C_{P_u}, C_{Q_i}) = S(C_{P_u}, C_{Q_i}) \cdot P_u Q_i^T
\]

where \( S(C_{P_u}, C_{Q_i}) \) is a similarity function to measure the similarity between user-cost vector \( C_{P_u} \) and item-cost vector \( C_{Q_i} \). Many existing similarity or distance functions can be used to perform \( S(C_{P_u}, C_{Q_i}) \), such as Pearson correlation coefficient and Euclidean distance. \( P_u \) and \( Q_i \) are user latent factors and item latent factors, respectively. The second one is based on the first model, with the difference that it uses a distribution to model user cost instead of representing it as a fixed 2-dimension vector as the first one.

### Emotion

In music recommendation, a number of works have tried to capture and model users’ emotion in music \[179,54,53\]. \[70\] proposes emotion aware music recommender system. It gives mathematical representation of emotion state transition. A set of emotion states are defined as \( E = \{e_1, e_2, \cdots, e_n\} \), where \( e_i \) denotes an emotion status, like “happiness” and ”sadness”. Then it describes a music feature at a specific moment \( t \) with an emotion state vector \( ES \):

\[
ES(t) = [es_1(t), es_2(t), \cdots, es_n(t)]
\]

where \( es_n(t) \) is a non-negative value at any specific moment \( t \) denoting the emotion strength. Then emotion state transition matrix (ESTM) is built by combining emotion state vectors of initial and final moments.

\[
ESTM = ES(t_{init})^T(E(t_{final}) - ES(t_{init}))
\]

\[
= \begin{bmatrix}
em_{1,1} & em_{1,2} & \cdots & em_{1,j} & \cdots & em_{1,n} \\
em_{2,1} & \cdots & \cdots & \cdots & \cdots & \cdots \\
\vdots & \ddots & \cdots & \cdots & \cdots & \cdots \\
em_{i,1} & \cdots & em_{i,j} & \cdots & \cdots & \cdots \\
\vdots & \ddots & \cdots & \cdots & \cdots & \cdots \\
em_{n,1} & \cdots & \cdots & \cdots & \cdots & em_{n,n}
\end{bmatrix}
\]
where $t_{\text{init}}$ denotes the initial moment, and $t_{\text{final}}$ is the final moment. $em_{ij}$ indicates the value of emotion influence from music. Ultimately, the system makes recommendation based on a user’s current emotion, desired emotion, and the music-to-ESTM mapping. Tags. Usually, tags consist of short descriptions about a certain aspect typical to an item. Some papers describe models to predict a item’s tags which help users to make a choice. For example, [50] uses AdaBoost [56] to predict music’s tags. Some works focus on leveraging tags to enhance recommender systems. For instance, [144] models social tags with 3-order tensors, which capture cubic correlations between users-tags-music items. It employs Higher Order Singular Value Decomposition (HOSVD) [104] [38], which generalizes SVD to multi-dimensional matrices, to learn latent factors of 3-dimensional space to make recommendation. With movie-mood information, [174] develops a recommender system. Taking movie mood as input, it first calculates mood-specific movie-movie similarity matrix, and then integrates it to matrix factorization models.

Demographic information. Demographic information is also helpful to alleviate data sparsity and cold start problems. Simply, we say features of users or items which do not change over time or relatively stable are demographic information. Such as birth of date, address, gender of a person, publisher of a book and producer of a movie. This information can be used to identify the types of users who like certain objects [153]. For example, ages, gender and education background are useful for tourism recommender system and advertisement agent. Publisher, category, layout of books can be used to calculate similarity between books [86].

We observe that context features are rarely employed to construct recommender system alone. Instead, researchers like to employ them to improve accuracy and alleviate data sparsity and cold start problems in existing systems.

### 4. Robustness of Recommender Systems

In recommender systems, people often assume that the rating information and other feedback provided by users are fair and honest. But in reality, this hypothesis may be wrong because malicious users can try to bias the recommendations for their own benefits. [149] is the first to show this sort of vulnerability of recommender systems. Now robustness, i.e. to deliver stable and accuracy personalized recommendations, is one of the most important security concerns for recommender systems.

Informally, robustness property measures the ability of preventing attackers from manipulating the output of recommender systems. Mathematically, we can define the robustness of recommendation algorithms as:

$$D : (f(R), f(R')) \rightarrow \tau, A : R \rightarrow R'$$

where $D$ is the distance function which can measure the influence of malicious attack, $f$ is recommendation algorithm, the output $\tau$ denotes the influence of the attacks. Strategic attacker $A$ inserts well-designed malicious profiles to recommender systems to result in output $R'$. Average shift prediction is often used to measure the influence of the attack to prediction recommendation. In this case, $D$ is defined by the difference between the output from $R$ and which from $R'$:

$$\tau = \frac{\sum|f(R) - f(R')|}{N}$$

where $N$ denotes the number of the user-item pairs in the test set. Average hit ratio is employed to measure the effectiveness of the attack for Top-N recommendation. It defines $D$ as an Edit distance:

$$\tau = \frac{\sum \text{Edit-Distance}(f(R), f(R'))}{N}$$

where $N$ is the number of the users in test set.

#### 4.1. Attack Models

Among all attacks against robustness, profile injection attacks against collaborative filtering methods have been studied extensively. In reality, the malicious users always try to achieve some particular recommendation bias. Commonly, there are two attack objectives.

- **Push attack**, which tries to secure positive recommendation for some items.
Nuke attack, which is designed to secure negative recommendation for some items.

Due to the specificity of profile injection attack, most of the attacks correspond to certain collaborative filtering methods. A\textsuperscript{3} compares the accuracy and attack influence among a variety of algorithms by inserting a set of hypothetical ratings to original dataset. Its distance function, called RMSS, is defined as:

$$RMSS = \sqrt{\frac{\sum_{(u,i) \in T} \Delta_{u,i}^2}{|T|}}$$

where $|T|$ is the number of rating pairs in test set $T$ and $\Delta_{u,i}$ denotes the difference of the prediction value on two datasets (the original dataset and the dataset with hypothetical ratings). Figure 4\textsuperscript{3} illustrates the accuracy (RMSE) and stability performance (RMSS). The methods located in the bottom left corner are the least accurate and least stable, while methods in the top right corner are the most accurate and most stable.

With well-designed ratings inserted into the malicious profiles, an attacker can perform Push attack or Nuke attack as expected. We present the attack procedures based on how the attackers manipulate the faked users profiles.

Random attack

Random attack\textsuperscript{A49} is designed to degrade the overall performance of recommender systems. Have a number of faked users, the attacker randomly selects some items rated by the users, and assigns them well-designed value, like maximum, minimum or any random value. Random attack is easy to execute and does not require much background knowledge. However, it is not particular effective against exist algorithms.

Average attack\textsuperscript{A49, A111} is more powerful against user based neighborhood methods. In this model, an attacker assigns a rating $r_{f,i}$ to item $i$, under the faked user $f$, with the item’s mean value: $r_{f,i} = \bar{r}_i$. It requires more background information than Random attack model. However, it falls short when applied to item based neighborhood methods.

Bandwagon attack,\textsuperscript{18, 20} is based on the observation that the items with high visibility, like items in the bestseller list, have a high possibility to be rated by many users. So malicious profiles containing such items would have more influence on the prediction or Top-N recommendation. These selected items are assigned maximum value and the targeted items are assigned with specific value to reach the attack purpose. Bandwagon attack is as effective as average attack (against user based neighborhood methods) without requiring much background knowledge. It is not effective when attacking item based neighborhood methods.

Instead of trying to influence as many users as possible, the attacks can be targeted to some specific users, such as students in university or the people older than 50. For this purpose, the attackers first discover the items which their target users prefer\textsuperscript{A40}. Then, they assign these items to the faked profiles with well-designed values to bias the recommendation. Overall, this kind of attacks requires little background knowledge and works well against both user based and item based neighborhood methods.

\textsuperscript{A191, A171} note that some users or items may have more influence than others, called power users (items). They introduce several methods to discover power users or power items. \textsuperscript{A191} indicates that an attacker can launch attacks by corrupting the power users (the cost for corrupting power users may be considerable high). Power user attack is effective to attack user based neighborhood and matrix factorization methods. Power item attack is effective against item based neighborhood methods.
4.2. Attack Detection

In profile injection attacks, the fake user profiles are often highly correlated and also have the ability to influence more users and items. Based on this observation, researchers have proposed a number of metrics to identify such users. The metrics are used to evaluate the attributes of users themselves and the correlation among users. For example, Number of Prediction Differences (NPD) indicates the number of prediction changes in the system after removing some specific users, and Degree of Similarity with Top Neighbors (DSTN) describes the average similarity weight with the Top-N neighbors [158]. Rating deviation from Mean Agreement (RDMA) and weighted degree of agreement (WDA) can reveal profiles which have low deviation from mean ratings value, but have high deviation from the mean of the attacked items [135]. By giving a threshold to each metric, users whose values are larger than the thresholds can be considered as malicious [31]. [135] finds that malicious users can highly correlate with each other as well as with genuine users. Hence, one problem with this method is the potential high false positive problem, namely some genuine users may be mistaken to be malicious. As a representative work of malicious users detection, [135] investigates the MovieLens dataset[4] and shows that the covariance between spam users is much lower than normal users. [135] employs Principal Component Analysis (PCA) [98] to detect spam users who have the minimum principal components. It shows that the PCA approach is effective to random attack, average attack, bandwagon attack, hybrid attack based on average attack and bandwagon attack, and also their variants based on obfuscation strategies. In the same direction, [137] uses SVD based approach to detect malicious profiles. Another popular yet different approach is employing classifier to class users into different classes and selecting the most suspicious class as malicious users. For example, [19] uses kNN classifier to find malicious users. It is clear that the attack effectiveness depends on the size of faked profiles. If a recommender system only accepts authentic profiles and allow them to post rating information then it will be very difficult for an attacker to mount an attack. Realizing this, many e-commerce websites only allow users to rate items which they bought. We observe that, due to the lacking of real dataset, researchers rarely use the time factor to enhance their detection methods. But in practice it’s reasonable to employ time as an important features because the attackers may only want to achieve their attack goals within a given time slot.

4.3. Algorithm Enhancement

When facing robustness attacks, [141] argues that model based algorithms are more robust than memory based algorithms. In contrast, some other researchers argue that model-based algorithms seem to be more robust is down to the fact that the proposed attacks have not hit the specific vulnerabilities of these algorithms, by illustrating diverse and obfuscated attacks on model-based recommender systems [29]. In the following, we introduce some representative works on improving robustness of existing recommender algorithms.

Reducing outliers’ influence is a popular approach to improve recommender algorithms’ robustness. [136] shows the vulnerability of matrix factorization is due to the cost function(least square), $\sum_{(u,i)}(r_{ui} - \hat{r}_{ui})^2$, which is sensitive to outliers. It then introduces M-estimator [88], which can alleviate outliers influence, with the cost function defined as:

$$\sum_{(u,i) \in K} \rho(e_{ui}) K^{-1}(e_{ui})$$

where $K$ is a empirical constant, $e_{ui}$ denotes $r_{ui} - \hat{r}_{ui}$. If $|e_{ui}| \leq K$ the value $\rho(e_{ui})$ is set to be 1, otherwise it is set to be $\frac{K}{|e_{ui}|}$. [30] argues that M-estimator does not improve recommender systems’ robustness significantly. Instead, it employs Least Trimmed Squares [162] to solve matrix factorization, with the cost function be defined as follows:

$$\sum_{i=1}^{k} e_i^2$$

where \( e_1 \leq e_2 \leq \cdots \leq e_n \), \( n \) is the total number of residuals and \( h < n \). The largest squared residuals are not used in the computation. Hence, it can mitigate the influence from outliers efficiently.

Trust network is another approach to enhance privacy and utility. Generally, we simply classify the proposed methods into two categories. Before going deep into the privacy-protection solutions, we briefly introduce some building blocks.

### 5. Privacy of Recommender systems

Privacy is another serious security concern of recommender systems, and it has gained enormous attention. Usually, privacy means that the recommender system should not leak any information beyond what can be inferred from the output and the service provider should not learn anything. Furthermore, some researchers demand that the output should not help an attacker to re-identify individuals and their attributes. In the following, we refer utility to the full performance (e.g. accuracy, effectiveness and so on) that normal recommender systems can provide without considering privacy. An ideal privacy-preserving solution should guarantee privacy without losing any utility. Unfortunately, there is always a tradeoff between privacy and utility. Generally, we simply classify the proposed methods into two categories.

- Data perturbation approach: the data owners perturb their own data and release them to each other. In this approach, the main purpose is to protect the privacy of individuals recorded in the datasets.

- Secure multiparty computation approach: recommendations are generated through a secure multiparty computation protocol.

#### 5.1. Building blocks

Before going deep into the privacy-protection solutions, we briefly introduce some building blocks.

- Garble circuits [196] [28] allow two parties, who have their own inputs \( x \) and \( y \) respectively, to evaluate any function \( f(x, y) \) without leaking any information about their inputs beyond what can be learned from the outputs. It has two stages: Garble and Evaluate. Assume a circuit \( C \) has \( n \) input wires. Garble produces the garbled circuit \( \tilde{C} \) and two labels \( \ell_i, b \), like encryption keys, for each input wire.

\[
(\tilde{C}, \{\ell_i, b\}_{i \in [n], b \in \{0, 1\}}) \leftarrow \text{Garble}(1^k, C)
\]

Every input corresponds to a value for each of the \( n \) input wires. Giving \( n \) of 2\( n \) input keys, then the circuit result can be evaluated with those keys.

\[
C(x) \leftarrow \text{Evaluate}(\tilde{C}, \{\ell_i, x_i\}_{i \in [n]})
\]

- Homomorphic encryption [61] allows operations to be done on ciphertexts and generate encrypted results. Formally, a homomorphic encryption scheme \( \epsilon \) is correct (a minimum request) for circuits in \( C_\epsilon \) if for any key-pair \((sk, pk)\) output by \( \text{KeyGen}_\epsilon(\gamma) \), any circuit \( C \in C_\epsilon \), any plaintext \( \pi_1, \pi_2, \ldots, \pi_n \) and any ciphertexts \( \Psi = <\psi_1, \psi_2, \ldots, \psi_n> \) with \( \psi_i \leftarrow \text{Encrypt}_\epsilon(pk, \pi_i) \), say if,

\[
\psi \leftarrow \text{Evaluate}_\epsilon(pk, C, \Psi)
\]

then,

\[
\text{Decrypt}_\epsilon(sk, \psi) \rightarrow C(\pi_1, \pi_2, \ldots, \pi_n)
\]

- Secure vector addition, mathematically, is described as:

\[
\exists : (X_1, X_2, \ldots, X_n) \rightarrow \sum_{i=1}^{n} X_i
\]

where \( \exists \) denotes the function to achieve secure computation. Each of \( n \) users has a vector of data \( X_i \in \mathbb{Z}^K \), each vector \( X_i \) has \( K \) coordinates, \( X_i = [X_{i1}, X_{i2}, \ldots, X_{ik}] \) and \( X_{ik} \) is integer. Secure vector addition can be implemented by cryptographic approach, like homomorphic encryption [22], and perturbation based approach such as [182] [65].
Secure scalar product is a widely used component to achieve secure computation. Formally, it is described as:

$$\mathcal{S} : (X, Y) \rightarrow \sum_i (X_i \cdot Y_i)$$

where $X \in \mathbb{Z}^K$ and $Y \in \mathbb{Z}^K$ are vectors. $X_i$ and $Y_i$ are integers. Algebraic solution \cite{181} and cryptographic solution \cite{201} are two popular approaches to achieve secure scalar product.

Secure comparison is a very important tool in computation, like generating Top-N recommendation. Given a pair of inputs $x$ and $y$, a secure function is designed to output a Boolean expression. For example, the output is 1, then we define $x \geq y$, otherwise $x < y$. We formulate the description as:

$$\mathcal{S} : (x, y) \rightarrow \{0, 1\}$$

Algebraic approaches, like \cite{148}, and cryptographic approaches, such as \cite{58}, are two main solutions to achieve secure comparison.

Differential privacy \cite{48} quantifies the privacy loss in a rigorous manner. Intuitively, it captures a single user’s influence on the output. Formally, \cite{48} defines it as: A randomized function $\mathcal{K}$ gives $\epsilon$-differential privacy if for all data sets $D_1$ and $D_2$ differing on at most one element, and all $S \subseteq \text{Range}(\mathcal{K}),$

$$\Pr[\mathcal{K}(D_1) \in S] \leq \exp(\epsilon) \times \Pr[\mathcal{K}(D_2) \in S]$$

$\epsilon$ is defined as privacy budget. It represents the extent of privacy that this model can guarantee. Two fundamental mechanisms to implement differential privacy are Laplace mechanism \cite{49} and Exponential mechanism \cite{134}. Laplace mechanism deals with numeric output and Exponential mechanism deals with non-numeric output.

5.2. Data Perturbation Approach

In this subsection, we review three different ways to perturb data in order to protect data privacy.

5.2.1. Additive data perturbation

Additive data perturbation has a long history in disclosing control of statistical databases. \cite{5} first introduces it to data mining area. Mathematically, additive data perturbation is described as:

$$Y = X + C$$

where $C$ is a noise matrix added to the original matrix $X$. In a common practice, each row of $C$ is generated independently from a certain distribution with mean $\mu$ (normally, it is shifted to zero) and variance $\delta^2$, such as Gaussian distribution and Uniform distribution.

In the case of recommender systems, a neighborhood based method is proposed in \cite{75}. The prediction is computed as follows:

$$\hat{r}_{ui} = \bar{r}_u + \delta_u \cdot \frac{\sum_{i \in R(i)} w_{uv} \cdot z_{vi}}{\sum_{i \in R(i)} w_{uv}}$$

and

$$w_{uv} = \sum_{k \in R(uv)} z_{uk} \cdot z_{vk}$$

where z-score $z_{ui}$ is defined as: $z_{ui} = \frac{r_{ui} - \bar{r}_u}{\delta_u}$. \cite{156} employs noise from uniform distribution to disguise the z-score matrix. Formally, $z'_{ui} = z_{ui} + c_{ui}$ where $c_{ui} \in [-\alpha, \alpha]$ is the noise value corresponding to user $u$ and item $i$. Then the data owner can publish $z'_{ui}$ without losing much privacy. By obtaining $z'_{ui}$, the following components, $\sum_{i \in R(i)} z_{uk} \cdot z_{ui} \approx \sum_{i \in R(i)} z'_{uk} \cdot z'_{ui}$ and $\sum_{u \in R(i)} z_{uk} \approx \sum_{u \in R(i)} z'_{{uk}}$, can be computed and sent back to the data owner. Thus, the data owner can get the final recommendation. Another example is on SVD based method \cite{157}. Recall that SVD factors a two-dimensional matrix into three sub sub-matrices as $R = U S V^T$, where $U$ denotes user latent factors, $V$ indicates item latent factors and $S$ is a diagonal matrix having all singular values of $R$. After filling the empty elements of $R$ with mean rating of users, matrix $R$ is normalized by converting rating to z-scores. Thus, the prediction can be presented as:

$$\hat{r}_{ui} = \bar{r}_u + \delta_u \cdot [U_k \sqrt{S_k(u)} \cdot V_k^T(i)]$$

where $k$ is the number of largest singular values used in the algorithm. \cite{167} uses a noise
matrix (following Uniform or Gaussian distributions) to disguise matrix $R$ to $R'$, and then computes $R' = U' S' V'^T$. It shows that the noise’s influence on $R^T R'$ can be converged to a negligible value. $S'$ and $V'$ can be estimated by calculating eigenvalues and eigenvectors of $R^T R'$, respectively. $U'$ can be estimated by computing $S^{-1} R V'$. With $U', S'$ and $V'$, users can obtain the final prediction.

**Figure 5: Wigner Semi-circle law**

**Challenges.** Most attacks against data perturbation solutions are based on two assumptions: The attackers can and only can access the perturbed dataset and the PDF of noise $C$ is public. Eigen-analysis is a popular approach to estimate original dataset. It is based on the observation that the eigenvalues of a random matrix are in a predictable distribution. Given a random matrix $C_{n \times n}$ whose rows are generated independently from a distribution with mean zero and variance $\sigma^2$, and let $n$ be a number large enough. Then, the PDF of the distribution of eigenvalues of $A + A'$ can be well estimated, shown in Figure 5. Describes a method to estimate original matrix $X$ based on Eigen-analysis. First, it normalizes matrix $Y$ to mean zero if it’s not. Then, it computes the standard covariance matrix $\Sigma_Y$ of $Y$, its eigenvalues $\gamma_1 \geq \gamma_2 \geq \cdots \geq \gamma_n$, and their corresponding normalized eigenvectors $v_1, v_2, \cdots, v_n$. Then, it chooses the $K$ eigenvectors associated with largest $K$ eigenvalues to form an eigen-matrix $V_Y$, denoted as: $[v_1, v_2, \cdots, v_K]$. Finally, it estimates $X$ to be

$$\hat{X} = V_Y V_Y^T Y$$

Bayesian based approach, like [87], considers both prior and posterior knowledge to estimate the original dataset. Let $p_X$ and $p_C$ denote the PDF of $X$ and $C$, respectively. Then, the estimation of $x$ can be obtained by maximizing

$$p_C(y-x)p_X(x), x \in \mathbb{R}^n$$

5.2.2. *Multiplicative data perturbation*

Multiplicative data perturbation leverages on data transformation methods to preserve privacy and keep data utility. Formally, it is described as:

$$Y = MX$$

while $M$ is the matrix chosen to transform the original data $X$. A number of methods have been introduced to construct matrix $M$ in [28]. For example, let $M$ be an orthogonal matrix $(M^T M = I)$. Then, it preserves Euclidean distance. Preserving distance or inner product is important to classification and collaborative filtering methods [27]. Take inner product as example, let $<X, Y> = X^T Y$ denotes inner product of $X$ and $Y$, and $M^T M = I$. So:

$$<M X, MY> = X^T M^T M Y = <X, Y>$$

For algorithms which can be implemented by inner product, multiplicative data perturbation can preserve similar accuracy to that based on original data.

Unfortunately, how well multiplicative data perturbation guarantees privacy is not clear, and it deserves further study. Some researchers find it vulnerable to background knowledge attack. If the attacker knows a portion of the original data set $X$, say $X_p$ and their corresponding part in perturbed data set $Y$, say $Y_p$. Suppose that $M$ is orthogonal, [121] shows how to use $X_p$ and $Y_p$ to obtain an estimation of $M$, denoted as $\hat{M}$. With $\hat{M}$, an estimation of $x_i$, denote as $\hat{x}_i$, can be computed as follows.

$$\hat{x}_i = \hat{M}^T y_i$$

5.2.3. *Differential Privacy*


- **Input Perturbation.** It is the same as [133] which adds noise to the data source directly.
- **In-process Mechanism.** The noise is applied in the middle of the execution of recommendation algorithms.
- **Output perturbation.** The perturbation is applied to the output. Take matrix factorization as an example, it adds noise
to the user latent factors and item latent factors [13].

[13] relies on Laplace mechanism to achieve differentially-private collaborative filtering recommender systems. It first analyzes the components of recommendation algorithms, like global effects: item’s rating summation: $MSum_i = \sum r_{ui}$, item’s rating number: $MCnt_i = \sum q_{ui}$ where $q_{ui}$ indicates if an item $i$ is rating by its corresponding user $u$; Covariance matrix: $\text{Cov}_{ij} = \sum r_{ui}r_{uj}$ and so on. Then, it analyzes the sensitivity of each components which is used to generate noise to satisfy differential privacy. Formally, define

$$K(X) = f(X) + \text{Laplace}(0, \delta)^d$$

say the mechanism $K$ guarantee $(\epsilon, \delta)$-differential if

$$\delta \geq \max_{D_1, D_2} \|f(D_1) - f(D_2)\|_1 / \epsilon$$

For example, to disguise item’s count number, it is: $GCnt = \sum q_{ui} + \text{Laplace}(0, \frac{1}{\epsilon})^d$. Since for count, $\max_{D_1, D_2} \|f(D_1) - f(D_2)\| = 1$.

[13] gives an example of differentially-private matrix factorization. When factoring matrix with SGD, the gradient is perturbed with noise which follows Laplace distribution in each iteration. [187], which uses Exponential mechanism, shows that sampling from the posterior distribution naturally achieves differential privacy. It proves that releasing one sample from posterior distribution $p(\theta|X^n)$ with any prior can guarantee 4B-differential privacy, where $\sup_{x, \theta \in \Theta} \|\log p(x|\theta)\| \leq B$, and $X$ is the data sample, $\theta$ is the hyperparameter, $sup$ is supremum. Based on this theorem, [123] successfully applies differential privacy to matrix factorization by sampling the estimated latent user factors and latent item factors. It’s an example of output perturbation.

Challenges. A major challenge of data perturbation approach is to achieve a desired privacy guarantee without losing much accuracy. Researchers have tried to seek a better balance between privacy and accuracy. On the other hand, researchers find that it is very difficult to achieve so called privacy guarantee under data perturbation methods. For example, Netflix provides a piece of highly sampled and perturbed data set for their competition. With a little background information, like several movies a user rated or a cross-correlated database, [145] successfully launches an attack to identify the records of known users and uncovering their political preferences. It’s important to be aware of which level of privacy is expected to achieve when applying differential privacy. User level and rating (attribute) level require different privacy budget which will affect accuracy directly. With data perturbation approach, single record can reach a certain level of privacy. But when these records are aggregated, many approaches can be employed to further mine the data and re-identify individual or attribute with nontrivial probability. For example, [35] demonstrates that classifiers, such as Naive Bayes classifier, can accurately infer ”private” attributes in differentially-private dataset.

5.3. Secure Multiparty Computation Approach

In this subsection, we review some representative privacy-preserving solutions with the flavor of security multi-party computation.

5.3.1. Secure Vector Addition based Solutions

[22] and [23] are two representative solutions which use secure vector addition to guarantee privacy for collaborative filtering recommender systems. Since they use the same methodology, we take the solution from [22] as an example. The scheme securely computes a certain aggregated value based on all users’ data in a distributed manner, and then publishes the aggregated information. With the public information, each user can locally compute his own personalized recommendations. In more detail, they construct a random $k$-dimensional linear space $A$ to best approximate the users rating matrix $R$ in a least squares manner, and let $AA^T = I$. Then they project $R$ to $A$ as $RA^T A$ and the cost function is defined as: $\epsilon = \text{tr}[(R - RA^T A)(R - RA^T A)^T]$, where $\text{tr}$ denotes trace. Since $\text{tr}(RR^T)$ is fixed, the optimization is to search for

$$A = \text{MAX}_{AA^T = I} \text{tr}(RA^T A R^T)$$

As an option, $A$ can be estimated by an iterative conjugate gradient algorithm with Polak-Ribiere recurrence [155] to maximize
A basic operation of SVD is matrix product. The security of neighborhood based methods can be implemented via a secure scalar product method, such as Pearson correlation coefficient, can be implemented via a secure scalar product method, or items, such as Pearson correlation coefficient, can be implemented via a secure scalar product method. Given horizontally (or vertically) partitioned data matrix $A = [X\ Y]^T$.

The computation of $XY^T$ and $YX^T$ can be achieved via secure matrix multiplication, and the summation during the calculation of $A$’s eigenvalues and eigenvectors of $AA^T$ and $A^TA$ can be achieved via secure vector (matrix) addition.

5.3.3. Garbled circuit based approach

Garbled circuit has not only been used to construct other secure computation components, but also been used to implement privacy-preserving algorithms directly. [146] [147] describes a framework based on garbled circuit to implement ridge regression, as shown in Figure 6.

![Figure 6: A secure computation framework based on garbled circuits](image)

CSP is the crypto service provider which constructs garbled circuits for algorithms, and Evaluator performs the garbled circuits with CSP. The users fully control their own data and submit the encrypted data to Evaluator. [146] implements a secure matrix factorization protocol in this framework. In more detail, the secure matrix factorization solution is shown in Figure 7.

1. Each user submits encrypted ratings to RecSys.
2. The RecSys masks ratings and sends them to CSP.
3. CSP encodes the decrypted ratings in the garbled circuits which implement the recommendation algorithm, and sends them to RecSys.

4. After evaluating the garbled circuits, RecSys can output the prediction to users.

![Figure 7: Secure matrix factorization](image)

5.3.4. Homomorphic encryption based approach

Similar to garbled circuits, homomorphic encryption schemes can be used to implement secure computation components. It has been employed to secure recommendation framework, such as [52][178][96]. For example, [178] proposes a distributed framework for computing recommendations without losing the participants’ privacy. In their framework, the rating information is encrypted and recommendation algorithms (they use neighborhood based method) is computed by employing homomorphic operations. In addition, a secure comparison protocol is employed to generate the final output.

Challenges. Secure multiparty computation based solutions suffer from several challenges. One challenge is computational complexity. Even though many researchers have tried to improve the performances of the building blocks. Some examples are HElib [5] and fastGC [6]. Nowadays, commercial recommender systems are often required to process massive data in real time. It will be a great challenge to guarantee privacy with cryptographic solutions. [178] has improved the situation a step further by mainly relying on a user’s friends to compute the recommendations. However, this will require the service provider to establish/maintain a social network for all its users and this task may not be too trivial. The other issue is the underlying imperfect security models, which often assume semi-honest attackers. For instance, [178] showed that the offline recommendation protocol proposed by [96] is vulnerable to key recover attacks.

6. Conclusion

In this paper we have surveyed the state of the art of recommender systems, and also the robustness and privacy issues and solutions. It is clear that there is a big gap between the system design and the privacy-preserving solution design. In the system design, researchers tend to include as many features as possible in an attempt to improve the recommendation performances. However, including more features incurs more privacy concerns because the service providers need to track users’ behavior to obtain these features in reality. The majority of existing solutions only aim to protecting the rating vectors for the users, and do nothing more. People may say that existing privacy-protection solutions such as anti-tracking schemes can be integrated to provide more privacy protection for the users. Unfortunately, it may not be so easy. We foresee a lot of efforts are required to fully understand the issue and find feasible solutions.

Acknowledgements

Both authors are supported by a CORE (junior track) grant from the National Research Fund, Luxembourg. Qiang Tang is also supported by an internal project from University of Luxembourg.


[38] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. A multilinear singu-


[57] Evgeniy Gabrilovich and Shaul


[133] Frank McSherry and Ilya Mironov. Differentially private recommender systems: building privacy into the net. In Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and...


[151] Dmitry Y Pavlov and David M Pennock. A maximum entropy approach to coll-


[198] Zhijun Yin, Liangliang Cao, Jiawei Han, Jiebo Luo, and Thomas S Huang. Diversified trajectory pattern ranking in geotagged social media. In SDM, pages 980–991. SIAM, 2011.


