Gaussian Process for Recommender Systems

Qi Liu[1], Enhong Chen[1], Biao Xiang[1], Chris H.Q. Ding[2], and Liang He[1]

[1] University of Science and Technology of China, Hefei, Anhui, China
{feiniaol,cheneh,bxiang,hshl05}@mail.ustc.edu.cn
[2] University of Texas, Arlington, TX 76019, USA
chqding@cse.uta.edu

Abstract. Nowadays, recommender systems are becoming increasingly important because they can filter noisy information and predict users' preferences. As a result, recommender system has become one of the key technologies for the emerging personalized information services. To these services, when making recommendations, the items' qualities, items' correlation, and users' preferences are all important factors to consider. However, traditional memory-based recommender systems, including the widely used user-oriented and item-oriented collaborative filtering methods, can not take all these information into account. Meanwhile, the model-based methods are often too complex to implement. To that end, in this paper we propose a Gaussian process based recommendation model, which can aggregate all of above factors into a unified system to make more appropriate and accurate recommendations. This model has a solid statistical foundation and is easy to implement. Furthermore, it has few tunable parameters, therefore it is very suitable for a baseline algorithm. The experimental results on the MovieLens data set demonstrate the effectiveness of our method, and it outperforms several state-of-the-art algorithms.

Keywords: Gaussian Process, Recommender Systems, Collaborative Filtering.

1 Introduction

With the rapid growth of information that is generated and collected from the Internet, sensors, wireless communications, etc, users have more and more choices and little time to make decisions. Examples of such decisions vary from the choice of movies and books in e-commerce to possible pick-up points for taxi drivers [8]. As a possible solution for such information overload problems, recommender systems can help users filter useless information and predict the items they may like or want to buy based on the preference model learned from their previous transactions, their demographic information, or the content of items, etc (see [1] for a survey). Many related techniques have been proposed and developed for designing effective recommender systems, such as the collaborative filtering methods [19], the content-based approaches [12] and the hybrid approaches [9].

Among them, collaborative filtering methods only require the information about user interactions, and do not rely on the content of items or user profiles, these methods have been widely implemented and studied [20]. Collaborative filtering based recommender systems usually assume that a given user will prefer an item, if other users with similar

H. Xiong and W.B. Lee (Eds.): KSEM 2011, LNAI 7091, pp. 56–67, 2011.

tastes and preferences liked or purchased this item in the past [1]. So far, collaborative filtering based recommender systems can be roughly categorized into memory-based and model-based methods.

The memory-based approaches usually first select a set of neighbor users for a given user, based on the entire collection of previously rated items by the users, then make recommendations based on the items that neighbor users like [17]. These memory-based methods are referred to as user-oriented memory-based approaches. In addition, an analogous procedure, which builds item similarity groups by the co-rating history, is known as item-oriented memory-based collaborative filtering [19].

On the other hand, the model-based approaches learn a model to predict possible rating of a user to each item. Recently, various model-based collaborative filtering approaches have been proposed, especially after the announcement of the Netflix prize contest [1] in October 2006. Those methods including graphical models [4,5], probabilistic approaches [11,23], dimensionality reduction techniques [13,14,18], etc. For those scoring systems, the recommendation algorithms first predict the possible rating value for each user-item pair, and then recommend high-rating items for each user.

However, traditional memory-based collaborative filtering methods can only take users' information (i.e., user-oriented) or items' information (i.e., item-oriented) into account, while the model-based methods are often too complex to implement (e.g., too many tunable parameters). To that end, in this paper, we construct a Gaussian process model (GPM) to learn user/item preferences. It is a unified model to take many factors into consideration, including the items' own qualities, the ratings correlations between items, the given user's rating preferences etc. Moreover, it has a strong **statistical foundation** and is **simple to implement**, thus, it is suitable for a baseline algorithm. In this model, we assume that user ratings for each item follow a Gaussian distribution (which is demonstrated in Section 3.2), thus the user profiles can be modeled by a Gaussian process approach. To verify the correctness and effectiveness of this model, we applied a unified algorithm, which is only based on the Gaussian process model, to the MovieLens data set, which is one of the standard data sets for evaluating recommender systems. The experimental results demonstrate that our algorithm is better than traditional collaborative filtering algorithms. Furthermore, our model does not require explicit parameter tuning (except for one regularization parameter).

The remainder of this paper is organized as follows. In Section 2, we describe our Gaussian process based recommendation model in detail. In Section 3, we show the experimental results in MovieLens data set.In Section 4, we introduce the related work. Finally, we conclude the paper in Section 5.

2 Gaussian Process Model

In this section, we first provide the definition of the recommendation problem. Then, we introduce how to make rating predictions based on Gaussian process model and its computational complexity. In addition, we describe the regularization problem for Gaussian process. Finally, we sum up each step together and illustrate the complete training process.

[1] http://www.netflixprize.com/

2.1 Problem Definition

Like many model-based recommender systems, in this paper, we formulate collaborative filtering mainly as a rating prediction problem. In the following, we note the input data matrix as R, the ratings of n users for p items. Each row of R denotes an item (a movie, a music, etc). The j-th column of R contains the rating scores that user $\mathbf{u}_j$ gives for each item (most of the users only rate a subset of the items). The task of the recommender system is to learn an information filtering procedure or a rating prediction function such that when a new user rates a few items, the recommender system automatically presents to the user many other selected items. These recommended items are chosen from the rest of the items that he/she has not yet rated, by considering the predicted ratings on them. As shown in Figure 1, for our recommendation model, the predicted ratings will be generated in the output.

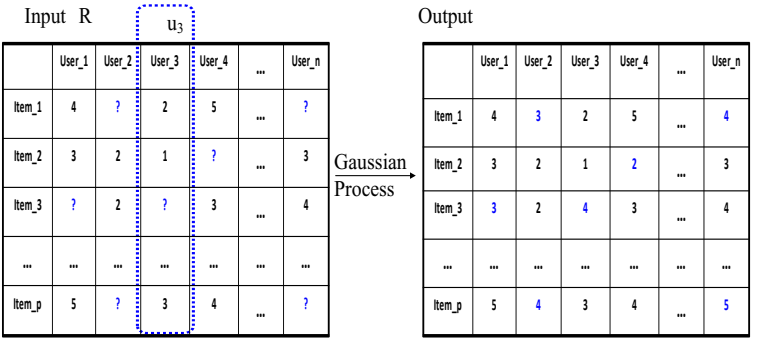


Fig. 1. The flowchart of the recommendation process, where the missing ratings in the input matrix are noted as '?'

In all, we approach this rating estimation problem from a model prediction point of view. Consider all ratings of a user would give on all items (i.e., suppose he/she had the opportunity to see all the items and has the time to rate all of them). This is called a user-profile, $\mathbf{u}$. In this way, the input rating data matrix R in Figure 1 can be viewed as $R = (\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_n)$, with lots of entries missing. Each profile $\mathbf{u}$ is a vector with size p.

We view a user-profile $\mathbf{u}$ as a vector of stochastic variables and assume their values follow a Gaussian distribution. We propose to use Gaussian process to model this user-profile $\mathbf{u}$. In the Gaussian process model, the user-profile $\mathbf{u}$ follows the multi-variate Gaussian distribution:

$$p(\mathbf{u}) \propto \exp\left[-\frac{1}{2}(\mathbf{u}-\bar{\mathbf{u}})^T \Sigma^{-1}(\mathbf{u}-\bar{\mathbf{u}})\right] \tag{1}$$

where the parameters for the model are $\Theta = (\bar{\mathbf{u}}, \Sigma)$. This means we must obtain the p-by-p rating covariance matrix Σ on the p items and the global mean $\bar{\mathbf{u}}$,which can be easily computed by the following equations:

$$\Sigma = \frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{u}_i-\bar{\mathbf{u}})(\mathbf{u}_i-\bar{\mathbf{u}})^T, \quad \bar{\mathbf{u}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{u}_i. \tag{2}$$

In these formulations, we can see that all entries in $R = (\mathbf{u}_1, \cdots, \mathbf{u}_n)$ are assumed to be *known*. However, in recommender system, there is a very large proportion of elements in R which are *unknown*. This is dealt with in the training process which is described in the following section.

2.2 Model Based Prediction

In this subsection, we show how to fill in the missing entries by a model based prediction. For clarity, suppose that we have already obtained $\bar{\mathbf{u}}$, Σ in this subsection, while we will discuss the computation of them in Section 2.5. Now given a new user $\mathbf{u}_i$, with partial ratings (the observed part) $\mathbf{u}_i^o$, the task is to predict the missing part $\mathbf{u}_i^m$.

At first, we re-order the index of the variables (e.g., movie items) in $\mathbf{u}_i$ such that the observed movie items which have been rated by user $\mathbf{u}_i$ are grouped together, and the movie items with missing ratings are grouped together. Thus, the corresponding notations can be represented by the following Equation:

$$\mathbf{u}_i = \begin{pmatrix} \mathbf{u}_i^o \\ \mathbf{u}_i^m \end{pmatrix},\ \bar{\mathbf{u}}_i = \begin{pmatrix} \bar{\mathbf{u}}_i^o \\ \bar{\mathbf{u}}_i^m \end{pmatrix}, \hat{\mathbf{u}}_i = \begin{pmatrix} \hat{\mathbf{u}}_i^o \\ \hat{\mathbf{u}}_i^m \end{pmatrix}, \Sigma = \begin{pmatrix} \Sigma_{oo} & \Sigma_{om} \\ \Sigma_{mo} & \Sigma_{mm} \end{pmatrix}. \tag{3}$$

where $\bar{\mathbf{u}}_i$ stores the global mean values for each movie item, and $\hat{\mathbf{u}}_i$ is a realization of the random vector variable for given user $\mathbf{u}_i$. In this paper, we can view $\hat{\mathbf{u}}_i^m$ as the predicted results for $\mathbf{u}_i^m$, and $\hat{\mathbf{u}}_i^o$ is always equal to $\mathbf{u}_i^o$. Correspondingly, covariance matrix Σ is re-ordered and decomposed into four sub-matrixes in a similar way.

Now, the task of predicting $\mathbf{u}_i^m$ with known $\mathbf{u}_i^o$ is identical to conditional Gaussian distribution [2]. Given the global mean $\bar{\mathbf{u}}$ and covariance matrix Σ, and the partial realizations of the vector of random variables, the Gaussian process model *asserts* that the probability distribution for the missing part $\mathbf{u}_i^m$ is still a Gaussian distribution:

$$\begin{aligned} p(\mathbf{u}_i^m) &\propto \exp\left[-\tfrac{1}{2}(\mathbf{u}_i^m - \hat{\mathbf{u}}_i^m)^T \hat{\Sigma}_{mm}^{-1}(\mathbf{u}_i^m - \hat{\mathbf{u}}_i^m)\right] \\ = exp[-\tfrac{1}{2}[(\mathbf{u}_i^m)^T \hat{\Sigma}_{mm}^{-1}\mathbf{u}_i^m &- (\mathbf{u}_i^m)^T \hat{\Sigma}_{mm}^{-1}\hat{\mathbf{u}}_i^m - (\hat{\mathbf{u}}_i^m)^T \Sigma_{mm}^{-1}\mathbf{u}_i^m + (\hat{\mathbf{u}}_i^m)^T \Sigma_{mm}^{-1}\hat{\mathbf{u}}_i^m]] \end{aligned} \tag{4}$$

where the parameters for this model are the *mean* $\hat{\mathbf{u}}_i^m$ and the *covariance* $\hat{\Sigma}_{mm}$.

In order to compute $\hat{\mathbf{u}}_i^m$ and $\hat{\Sigma}_{mm}$, first consider the quadratic form in the exponent of the Gaussian distribution of all the items:

$$\begin{aligned} p(\mathbf{u}_i) &\propto \exp\left[-\tfrac{1}{2}(\mathbf{u}_i - \bar{\mathbf{u}}_i)^T \Sigma^{-1}(\mathbf{u}_i - \bar{\mathbf{u}}_i)\right] \\ = \exp\left[-\tfrac{1}{2}[(\mathbf{u}_i^o - \bar{\mathbf{u}}_i^o)^T \wedge_{oo} (\mathbf{u}_i^o - \bar{\mathbf{u}}_i^o) + (\mathbf{u}_i^o - \bar{\mathbf{u}}_i^o)^T \wedge_{om} (\mathbf{u}_i^m - \bar{\mathbf{u}}_i^m)\right. \\ \left. +(\mathbf{u}_i^m - \bar{\mathbf{u}}_i^m)^T \wedge_{mo} (\mathbf{u}_i^o - \bar{\mathbf{u}}_i^o) + (\mathbf{u}_i^m - \bar{\mathbf{u}}_i^m)^T \wedge_{mm} (\mathbf{u}_i^m - \bar{\mathbf{u}}_i^m)]\right] \end{aligned}$$

where $\wedge$ is Σ^{-1}, and can be represented as $\begin{pmatrix} \wedge_{oo} & \wedge_{om} \\ \wedge_{mo} & \wedge_{mm} \end{pmatrix}$. If we apply this equation to the conditional Gaussian distribution $p(\mathbf{u}_i^m|\mathbf{u}_i^o)$ and pick out all terms that are second order in $\mathbf{u}_i^m$, we have $-\frac{1}{2}(\mathbf{u}_i^m)^T \wedge_{mm} \mathbf{u}_i^m$. Meanwhile, as shown in Equation (4), for Gaussian distribution, the matrix of coefficients entering in the second order term in $\mathbf{u}_i^m$ should be the inverse covariance matrix $\hat{\Sigma}_{mm}^{-1}$, thus $\hat{\Sigma}_{mm} = \wedge_{mm}^{-1}$.

Further, consider all the terms in the exponent that are linear in $\mathbf{u}_i^m$, we have:

$$(\mathbf{u}_i^m)^T[\wedge_{mm}\bar{\mathbf{u}}_i^m - \wedge_{mo}(\mathbf{u}_i^o - \bar{\mathbf{u}}_i^o)]$$

where we have used $\wedge_{om}^T = \wedge_{mo}$. While for Equation (4), the coefficient of $\mathbf{u}_i^m$ in this expression equals to $(\mathbf{u}_i^m)^T\hat{\Sigma}_{mm}^{-1}\hat{\mathbf{u}}_i^m$, and hence

$$\hat{\mathbf{u}}_i^m = \bar{\mathbf{u}}_i^m - \wedge_{mm}^{-1}\wedge_{mo}(\mathbf{u}_i^o - \bar{\mathbf{u}}_i^o)$$

Then, using the definition $\begin{pmatrix}\Sigma_{oo} & \Sigma_{om}\\ \Sigma_{mo} & \Sigma_{mm}\end{pmatrix}^{-1} = \begin{pmatrix}\wedge_{oo} & \wedge_{om}\\ \wedge_{mo} & \wedge_{mm}\end{pmatrix}$ and making use of the inverse of a partitioned matrix [2], we can get:

$$\hat{\mathbf{u}}_i^m = \bar{\mathbf{u}}_i^m + \Sigma_{mo}\Sigma_{oo}^{-1}(\hat{\mathbf{u}}_i^o - \bar{\mathbf{u}}_i^o) \tag{5}$$

$$\hat{\Sigma}_{mm} = \Sigma_{mm} - \Sigma_{mo}\Sigma_{oo}^{-1}\Sigma_{om} \tag{6}$$

We emphasize again that $\hat{\mathbf{u}}_i$ is a realization of $\mathbf{u}_i$, and $\hat{\mathbf{u}}_i^o$ equals to $\mathbf{u}_i^o$. After learning (through the training process described in Section 2.5) the model parameters $(\bar{\mathbf{u}}, \Sigma)$ and given the input $\hat{\mathbf{u}}_i^o$, the prediction results for user $\mathbf{u}_i$'s missing part is $(\hat{\mathbf{u}}_i^m, \hat{\Sigma}_{mm})$, where $\hat{\mathbf{u}}_i^m$ is the predicted result for $\mathbf{u}_i^m$.

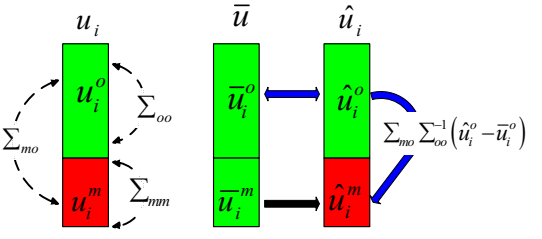


Fig. 2. The implication of the predicting Equation(5), and the meanings of the rating value influential factors

Implications. From an application point of view, it is worth understanding the each part's implication of the prediction Equation (5). Actually, this equation shows that each rating prediction is made by considering two values: the global mean (i.e., $\bar{\mathbf{u}}_i^m$) and its modification (i.e., $\Sigma_{mo}\Sigma_{oo}^{-1}(\hat{\mathbf{u}}_i^o - \bar{\mathbf{u}}_i^o)$). Here $\bar{\mathbf{u}}_i^m$ stands for the quality of the movie item, which can be concluded from the ratings of other users. $\Sigma_{mo}\Sigma_{oo}^{-1}(\hat{\mathbf{u}}_i^o - \bar{\mathbf{u}}_i^o)$ can be seen as the impact of $\mathbf{u}_i$'s rating preferences. Specifically, this impact is modeled by the difference between $\mathbf{u}_i$'s rating preference and all users' rating preferences, and this difference is weighted by the rating correlations between the movie items. These influential factors are illustrated in Figure 2.

2.3 Computational Complexity

A key observation regarding the computational efficiency of the Gaussian process model for recommender system is the following. At first glance, the inverse of Σ is computationally expensive because Σ is a p-by-p matrix, where p could be tens of thousands

[2] For more detailed inference on Equation (5) and Equation (6), please refer to Ref. [2].

(e.g., $p = 18,000$ for the Netflix system), and the computational complexity of this matrix inversion is $O(p^3)$. However, in real-world recommender systems, we never need to compute or evaluate Σ^{-1}. All we need is to compute Σ_{oo}^{-1} efficiently (please refer to Equation(5)). Since the number of movie items that each user can rate may only vary from several dozens to 200, which means the size of Σ_{oo} is at most 200-by-200. Inverting a matrix of this size is computationally trivial, thus the prediction can be done very quickly and efficiently.

2.4 Regularized Gaussian Process

There always exist many random factors and noises for such diverse collection of users of a recommender system—one user may give many high scores to his/her favorite movies and does not give scores to the ones that he/she does not appreciate; some other users may only give scores for movies they dislike while not bother giving scores to the movies they like.

For these reasons, in this subsection we propose to use regularization in the Gaussian process. More specifically, we believe that the covariance matrix for items from the computed values of Equation (2) could have significant noise, and a regularization is proposed on it. As the following, when predicting $(\hat{\mathbf{u}}_i^m, \hat{\Sigma}_{mm})$, where in Equations (5 and 6), we replaced the covariance matrix from the computed values of Equation (2) by

$$\Sigma \leftarrow \Sigma + \beta * I$$

where I is the $p \times p$ identity matrix and $\beta > 0$ is a parameter. This type of regularization is often used in statistics and is sometimes called *Ridge* regularization. In fact, this kind of noise has been manifested by many other works. Besides the ridge regularization that we propose to use in this paper, other regularization methods may also be applicable [21].

Parameter Setting. Since Σ is a covariance matrix, it is real symmetric and positive semi-definite. We can write

$$\Sigma = V \overline{\wedge} V^T = \sum_{i=1}^{p} \lambda_i v_i v_i^T$$

where $V = [v_1, v_2, ...v_p]$ and $\overline{\wedge} = diag\{\lambda_1, \lambda_2, ...\lambda_p\}$ [3] are the eigenvector matrix and eigenvalue diagonal matrix of Σ, respectively. Then,

$$\Sigma^{-1} = \sum_{i=1}^{p} \frac{1}{\lambda_i} v_i v_i^T, \qquad (\Sigma + \beta * I)^{-1} = \sum_{i=1}^{p} \frac{1}{\lambda_i + \beta} v_i v_i^T \tag{7}$$

From Equation 7, we can see that the eigenvectors with smaller eigenvalues in Σ will contribute significantly to its inverse Σ^{-1}, while not after the ridge regularization. To prevent the eigenvectors with smaller eigenvalues contributing too much to Σ^{-1}, we should choose a proper β value. In our study, we set β equaling to λ_{20}, which is the 20th largest eigenvalue of Σ, so that **effectively** eigenvectors with eigenvalues less than λ_{20} will have nearly equal contributions to $(\Sigma + \beta * I)^{-1}$. In practice, eigenvalues decrease very fast at the beginning, and then slowly. Thus choosing β=λ_{20} or β=λ_{30} makes very little difference.

[3] In this paper, we note that $\lambda_1 \geq \lambda_2 ... \geq \lambda_p \geq 0$.

2.5 Training

In this subsection, we illustrate the complete training process of our model, including the computation of model parameters $\Theta = (\bar{\mathbf{u}}, \Sigma)$ and the final prediction for $\hat{\mathbf{u}}_i^m$.

Initialization. The first step in training process is to initialize the input matrix R by filling its missing ratings. We consider two simple approaches here (more sophisticated methods can be designed). (1) Item-based initialization. We compute the mean value of each item based on those already filled entries. The missing entries of this item are filled with this mean value. (2) User-based initialization. We compute the mean value of each user based on existing filled entries. The missing entries of this user are filled with this mean value.

Process. The training process is the iteration on the following two steps:

1. Given the *current* estimation of the missing part in R, we compute the updated model parameters $\Theta = (\bar{\mathbf{u}}, \Sigma)$, by the following Equation:
$$\Sigma = \frac{1}{n-1}\sum_{i=1}^{n}(\hat{\mathbf{u}}_i - \bar{\mathbf{u}})(\hat{\mathbf{u}}_i - \bar{\mathbf{u}})^T + \beta * I, \qquad \bar{\mathbf{u}} = \frac{1}{n}\sum_{i=1}^{n}\hat{\mathbf{u}}_i.$$
2. Given the current estimation of the model parameters Θ, we use conditional Gaussian distribution formula Equation (5) to predict a newer and better estimate of $\mathbf{u}_i^m$ for all users $\mathbf{u}_i$ (the newer $\hat{\mathbf{u}}_i^m$).

The above two steps are repeated until convergence or reaching a pre-specified maximum number of iterations, for example 10 iterations.

3 Experimental Evaluation

In this section, we present the experimental results to evaluate the performance of our Gaussian process model based recommender system. Specifically, we demonstrate: (1) the observation of our assumption on the Gaussian distribution of user ratings; (2) a performance comparison between our model and the benchmark methods.

3.1 Experimental Setup

Experiments were performed on the MovieLens data set, which is a benchmark data set for evaluating recommender systems [3]. This data set contains 100, 000 ratings from 943 users for 1, 682 movies, and each user has rated at least 20 movies. The rating values vary from 1 to 5.

We do the following cross-validation. For each user's ratings, we randomly selected some percentage of the ratings for training and the remaining ones for testing. In total, we constructed 5 pairs of training and test sets: 90:10, 80:20, 70:30, 60:40 and 50:50. For example, 90:10 split means 90% ratings are used as the training data and the remaining 10% ratings serve as the test data.

Evaluation Measures. We use the *mean absolute error* (MAE) and the *root mean squared error* (RMSE) to evaluate the quality of the results. Both of them are widely

used for the purpose of evaluating the rating effectiveness. If $\hat{u}_i^{T_i}$ is the test part for user $\mathbf{u}_i$, with its size T_i, then these measures can be defined as follow:

$$MAE = \frac{\sum_{i=1}^{n}\sum_{t=1}^{T_i}|\hat{u}_i^t - u_i^t|}{|\sum_{i=1}^{n} T_i|}, \qquad RMSE = \sqrt{\frac{\sum_{i=1}^{n}\sum_{t=1}^{T_i}|\hat{u}_i^t - u_i^t|^2}{|\sum_{i=1}^{n} T_i|}}$$

Benchmark Methods. In order to demonstrate the effectiveness of our Gaussian process based recommendation model, we compare it with many benchmark methods. Specifically, we implement two widely used item-based [19] and user-based [17] (i.e., item/user-oriented memory-based) collaborative filterings, as well as a SVD model based method named regularized SVD [6]. For the item-based collaborative filtering, adjusted cosine method is used to compute item similarity and weighted sum is used to make predictions [19]. For the user-based method, Pearson's correlation is chosen to compute user similarities [17]. For the regularized SVD, we set the parameters similar with Simon Funk, except for the number of features K, which was not mentioned in Ref. [6]. To that end, we train K from 20 to 200 steps by 20, and finally we get its best performance at K equaling to 120. All these three benchmark methods are the state-of-the-art collaborative filtering algorithms, and they are widely used for baselines.

3.2 Results and Discussions

First, we demonstrate our assumption on the Gaussian distribution of user ratings, which is a common practice in many previous works, including Ref. [6]. We tested our assumption on the MovieLens data set, and the corresponding results are shown in Figure 3 and Figure 4.

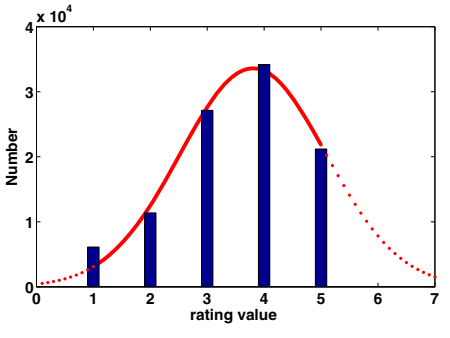


Fig. 3. The Gaussian distribution of all user rating values

Figure 3 illustrates the distribution of all user ratings in the MovieLens data set. We can see that the distribution of different rating values can be well fitted by a Gaussian distribution. [4] To demonstrate the Gaussian assumption for each variable (i.e., movie), we choose the first three movies in this data set as an example, and their rating values

[4] All the Gaussian distributions in this paper are obtained by making use of the Matlab curve-fitting toolbox cftool, and the R-Square of this Gaussian distribution is equal to 0.9772.

Table 1. Experimental results for different algorithms

(a) (MAE)

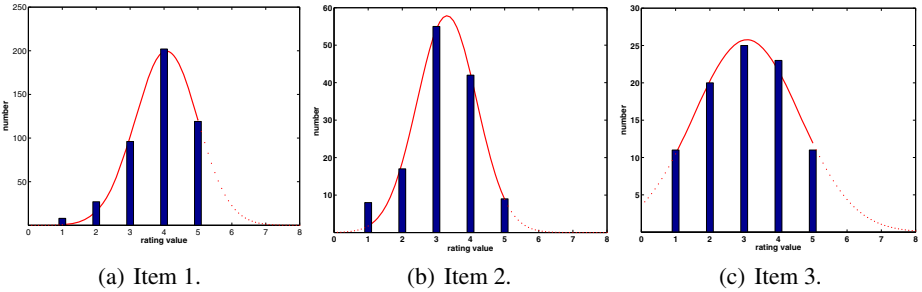
Alg. \ Split		50 : 50	60 : 40	70 : 30	80 : 20	90 : 10
Gaussian Process Init	Item-based	0.8199	0.8160	0.8166	0.8120	0.8223
	User-based	0.8348	0.8347	0.8346	0.8320	0.8448
Gaussian Process	Item-based Init	0.7491	0.7361	0.7249	0.7219	0.7191
	User-based Init	**0.7361**	**0.7266**	**0.7153**	**0.7112**	**0.7095**
Traditional CF	Item-based	0.7581	0.7520	0.7502	0.7487	0.7552
	User-based	0.7561	0.7514	0.7446	0.7412	0.7465
Regularized SVD		0.7517	0.7489	0.7472	0.7406	0.7395

(b) (RMSE)

Alg. \ Split		50 : 50	60 : 40	70 : 30	80 : 20	90 : 10
Gaussian Process Init	Item-based	1.0282	1.0231	1.0229	1.0190	1.0289
	User-based	1.0409	1.0397	1.0405	1.0367	1.0539
Gaussian Process	Item-based Init	0.9559	0.9399	0.9244	0.9207	0.9207
	User-based Init	**0.9359**	**0.9243**	**0.9080**	**0.9040**	**0.9006**
Traditional CF	Item-based	0.9673	0.9565	0.9526	0.9520	0.9601
	User-based	0.9619	0.9569	0.9459	0.9425	0.9465
Regularized SVD		0.9575	0.9477	0.9378	0.9339	0.9149

are described in Figure 4. We can see that the ratings for each movie can also be well fitted by a Gaussian distribution, and different movies have different means and variances. Though the Gaussian assumption is very simple and practical, we note that there are still many limitations for it. For example, for many reasons (e.g., for lack of enough ratings), there may exist movies whose rating distributions are not obviously Gaussian. What's more, the ratings in many data sets may not be rated as we expected, and for these situations, incorrect assumptions will lead to biased prediction. Thus, more sophisticated and appropriate models should be designed [16].

Then, we present an effectiveness comparison between our Gaussian process model and three benchmark approaches. The results of each method with respect to

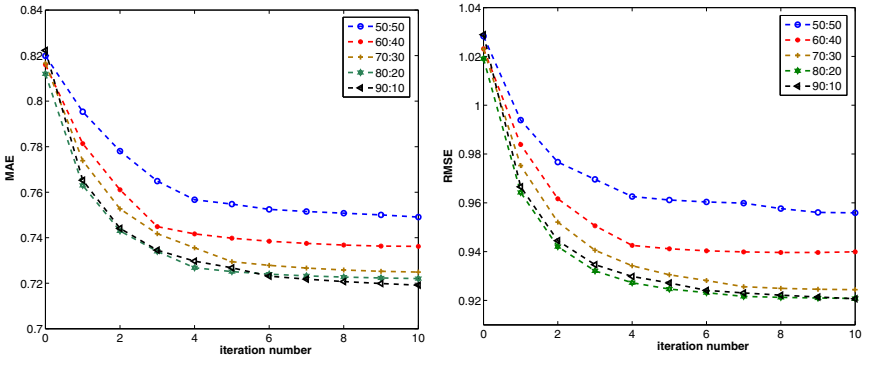


(a) Item 1. (b) Item 2. (c) Item 3.

Fig. 4. The Gaussian distribution of the ratings for the first three items

different splits are illustrated in Table 1. In this table, "Gaussian Process Init" rows list MAE/RMSE measures on the results after the **Initialization** step (Section 2.5); while "Gaussian Process" rows list the results after the iteration **Process** step.

From Table 1 we can see that since Gaussian process based algorithm can aggregate many rating factors, it clearly outperforms the traditional item-based and user-based collaborative filtering algorithms (Traditional CF) in each split with a significant margin, and it also performs better than the regularized SVD, regardless of which similarity measure has been chosen. On the other hand, it is interesting to find that the Gaussian process algorithm with user-based initialization performs better than the one with item-based initialization, and this relationship is similar to the results listed in the "Traditional CF" rows. Further more, Figure 5 shows the effect of the number of iterations on the performances of our Gaussian process algorithm with item-based initialization, from which we can see that each iteration of the training process does enhance the performance of rating prediction and all the rating curves converge after only a few steps.



(a) MAE results. (b) RMSE results.

Fig. 5. The experimental results according to different number of iterations for each split

In summary, considering the simplicity of the Gaussian process based recommendation model (without tunable parameters except β which is always to the 20^{th} largest eigenvalue of Σ), these results are very encouraging.

4 Related Work

Recommender systems emerged as an independent research area in the mid-1990s when researchers started working on recommendation problems that explicitly rely on the ratings structure [1]. In its most common formulation, the recommendation problem is reduced to the problem of estimating ratings for the items that have not been seen by a user. Meanwhile, some other approaches consider the recommendation problem as either a classification problem [10] or a ranking problem [15].

As can be seen from this paper, our Gaussian process model based recommender system can be classified as one of the model-based rating estimation oriented collaborative filtering methods. In our model, the user ratings on each item are assumed to follow a

Gaussian distribution. Actually, there are many other recommendation scenarios, where Gaussian distribution is chosen as a prior or hyper distribution for modeling data. For example, Hoffman presents an expectation maximization (EM) algorithm for collaborative filtering that estimates latent classes with Gaussian probability distributions [11]; Salakhutdinov et al. propose a Probabilistic Matrix Factorization (PMF) based collaborative filtering model by introducing Gaussian noise to observed user ratings [18], and Ge et al. successfully extend this PMF model for recommending travel packages by considering both the travel cost and the tourist's interests [7].

To the best of our knowledge, among these recommendation models where the assumption of Gaussian distribution is considered, the models proposed in Ref. [21] and Ref. [22] are the most closely related to our work. In Ref. [21], Schwaighofer et al. propose the method of learning Gaussian process covariance functions for multi-task learning problems from a hierarchical bayesian point of view, and Umyarov et al. extend this collaborative filtering model by incorporating externally specified aggregate rating information [22]. Though we are based on the same Gaussian distribution assumption, in this paper, we explored the implications of this type of model from an application point of view, and consequently, we designed a novel training process which makes our method easier for implementation and understanding. However, we should note that, as experimentally observed in Ref. [16], rating data in different recommendation applications may have different properties, and we should observe their characteristics carefully before any assumptions are made.

5 Concluding Remarks

In this paper, we propose a Gaussian process model for building an effective collaborative filtering based recommender system. This model can aggregate many factors, such as items' qualities, items' correlations, and users' rating preferences, which are all very important for personalized information services, into a unified system. The experimental results demonstrate that this algorithm is effective, and it outperforms many state-of-the-art methods including two traditional collaborative filterings and a SVD based method. What's more, this algorithm has a solid statistical foundation, is easy to implement and has little hassle of tuning parameters. Thus, it is suitable for a baseline algorithm. We plan to further refine the model and do more experiments on other data sets like Netflix.

Acknowledgments. This research was supported in part by National Natural Science Foundation of China (Grant No. 61073110), the Key Program of National Natural Science Foundation of China (Grant No. 60933013), the research fund for the Doctoral Program of Higher Education of China (Grant No. 20093402110017), the National Major Special Science & Technology Project (Grant No. 2011ZX04016-071).

References

1. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Trans. Knowl. and Data Eng. (TKDE) 17, 734–749 (2005)

2. Bishop, C.M.: Pattern recognition and machine learning, vol. 4, ch. 2. Springer, New York (2006)
3. GroupLens Research (2007), http://www.grouplens.org/node/73#attachments
4. Ding, C., Jin, R., Li, T., Simon, H.D.: A learning framework using Green's function and kernel regularization with application to recommender system. In: ACM SIGKDD, pp. 260–269 (2007)
5. Fouss, F., Pirotte, A., Renders, J.-M., Saerens, M.: Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. IEEE Trans. Knowl. Data Eng. (TKDE) 19(3), 355–369 (2007)
6. Funk, S.: Netflix update: Try this at home (2006), http://sifter.org/~simon/journal/20061211.html
7. Ge, Y., Liu, Q., Xiong, H., Tuzhilin, A., Chen, J.: Cost-aware travel tour recommendation. In: ACM SIGKDD, pp. 983–991 (2011)
8. Ge, Y., Xiong, H., Tuzhilin, A., et al.: An energy-efficient mobile recommender system. In: ACM SIGKDD, pp. 899–908 (2010)
9. Gunawardana, A., Meek, C.: A unified approach to building hybrid recommender systems. In: ACM RecSys, pp. 117–124 (2009)
10. Herlocker, J.L., Konstan, J.A., Terveen, L.G., Riedl, J.T.: Evaluating collaborative filtering recommender systems. ACM Transactions on Information Systems (TOIS) 22(1), 5–53 (2004)
11. Hofmann, T.: Latent semantic models for collaborative filtering. ACM Transactions on Information Systems (TOIS) 22(1), 89–115 (2004)
12. Kim, J.W., Lee, B.H., Shaw, M.J., Chang, H.-L., Nelson, M.: Application of decision-tree induction techniques to personalized advertisements on internet storefronts. Int. J. Electron. Commerce 5(3), 45–62 (2001)
13. Koren, Y.: Collaborative filtering with temporal dynamics. In: ACM SIGKDD, pp. 447–456 (2009)
14. Kurucz, M., Benczur, A.A., Csalogany, K.: Methods for large scale SVD with missing values. In: ACM KDDCup 2007, pp. 31–38 (2007)
15. Liu, Q., Chen, E., Xiong, H., Ding, C.H.Q.: Exploiting user interests for collaborative filtering: interests expansion via personalized ranking. In: ACM CIKM, pp. 1697–1700 (2010)
16. Marlin, B.M., Zemel, R.S.: Collaborative prediction and ranking with non-random missing data. In: ACM RecSys, pp. 5–12 (2009)
17. Paul, R., Neophytos, I., Mitesh, S., Peter, B., John, R.: GroupLens: an open architecture for collaborative filtering of netnews. In: ACM CSCW, pp. 175–186 (1994)
18. Salakhutdinov, R., Mnih, A.: Probabilistic matrix factorization. In: NIPS, vol. 20, pp. 1257–1264 (2008)
19. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Item-based collaborative filtering recommendation algorithms. In: WWW, pp. 285–295 (2001)
20. Schafer, J.B., Frankowski, D., Herlocker, J., Sen, S.: Collaborative Filtering Recommender Systems. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.) Adaptive Web 2007. LNCS, vol. 4321, pp. 291–324. Springer, Heidelberg (2007)
21. Schwaighofer, A., Tresp, V., Yu, K.: Learning Gaussian process kernels via hierarchical Bayes. In: NIPS, vol. 17, pp. 1209–1216 (2005)
22. Umyarov, A., Tuzhilin, A.: Improving collaborative filtering recommendations using external data. In: IEEE ICDM, pp. 618–627 (2008)
23. Wu, H., Wang, Y., Cheng, X.: Incremental probabilistic latent semantic analysis for automatic question recommendation. In: ACM RecSys, pp. 99–106 (2008)