LTMF: Local-Based Tag Integration Model for Recommendation

Deyuan Zheng[1], Huan Huo[1(✉)], Shang-ye Chen[2], Biao Xu[1], and Liang Liu[1]

[1] University of Shanghai for Science and Technology, Shanghai 200093, China
huo_huan@yahoo.com
[2] School of Information and Technology, Northwest University, Xian 710127, China

Abstract. There are two primary approaches to collaborative filtering: memory- based and model-based. The traditional techniques fail to integrate with these two approaches and also can't fully utilize the tag features which data contains. Based on mining local information, this paper combines neighborhood method and matrix factorization technique. By taking fuller consideration of the tag features, we propose an algorithm named LTMF (Local-Tag MF). After the real data validation, this model performs better than other state-of-art algorithms.

1 Introduction

As one of the two main collaborative filtering techniques, memory-based method is to calculate the similarity to find the neighbors of users or items and produce clusters to complete the recommendation. The model-based method typically adopts the view called latent factor model, which assumes that the users or items are composed of latent factors. By extracting the latent factor from user vector or item vector through mathematic method, we can find the user with matching items to complete the recommendation.

Traditional memory-based model and model-based method barely can be combined. Moreover, most algorithms do not fully consider tags, which can aggravate the sparsity problem of rating matrices. Also, without fully using the tags, it will result the "cold starting" problem. We must explore the local structure and similarity to analyze the connections between data that not explicit annotated. [1] considered the tag features, and [2] introduced the mix-recommendation. However, these new techniques still fail to fully take account of data?s potentials and latent information.

This paper takes full account of data information, builds model from two sides. First exploring the implicit local structure by clustering algorithm, then making use of data tags to build new matrices. At last, we propose LTMF model which is better than above.

2 Related Work

There are many mature research about latent factor on collaborative filtering. Memory- based algorithm [3,4] mainly calculated the similarity between users and articles and then predict the missing values based on the existing rating matrices. [5] made a further study on sparse matrix completion algorithm. [6] proposed some methods consider KNN to this kind of problem. Among

S. Guo et al. (Eds.): CollaborateCom 2015, LNICST 163, pp. 296–302, 2016.
DOI: 10.1007/978-3-319-28910-6_27

model-based algorithms, [7] made an improvement on SVD which is called LFM algorithm and applied it on collaborative filtering. [8] proposed PMF by adding probabilistic dimension into LFM.

2.1 Probabilistic Matrix Factorization

The framework of PMF algorithm is shown as formula 1:

$$p(R|U,V,\sigma_R^2) = \Pi_{u=1}^{M}\Pi_{i=1}^{N}[N(R_{u,i}|U_u^T V_i,\sigma_R^2)]^{I_{u,i}^R} \tag{1}$$

$N(x|\mu,\sigma^2)$ denotes Probability Density Function of normal distribution with μ as the mathematical expectation of rating matrix and σ^2 as the variance. $I_{u,i}^R$ is the indicator that equals to 1 if user u rated item i and equals to 0 otherwise. Formulas 2 and 3 assume that the mathematical expectation of user and item is 0, and the variances are σ_U^2 and σ_V^2 respectively:

$$p(U|\sigma_U^2) = \Pi_{u=1}^{M}N(U_u|0,\sigma_U^2 I) \tag{2}$$

$$p(U|\sigma_V^2) = \Pi_{i=1}^{N}N(V_i|0,\sigma_V^2 I) \tag{3}$$

The loss function is defined as:

$$E = \frac{1}{2}\sum_{u=1}^{M}\sum_{i=1}^{N}I_{ui}^R(R_{ui} - U_u^T V_i)^2 + \frac{\lambda_U}{2}\sum_{u=1}^{M}||U_u||_F^2 + \frac{\lambda_V}{2}\sum_{i=1}^{N}||V_i||_F^2 \tag{4}$$

where $\lambda_U = \sigma_R/\sigma_U$, $\lambda_V = \sigma_R/\sigma_V$. The core of PMF and LFM is quite similar. But considering the randomness of latent factor, PMF always outperforms LFM. PMF is generally trained by SGD algorithm like LFM.

2.2 Other Work

Many data with tags are embedded on the Internet. [9] proposed a tag-based algorithm for recommendation. [10] used the item's type information to solve the problem of sparse and cold start. [1] proposed Tag-LFM to construct a new matrix with taking full advantage of tag features. [11] went a step further with a MF model based on local similarity named SBMF (Similarity-Based Matrix Factorization) algorithm. Most of these algorithms refer to matrix construction. A typical technique is shown in Fig. 1.

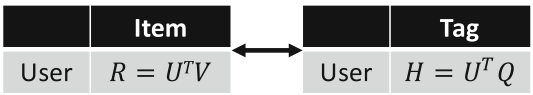

Fig. 1. New matrix construction

We assume that H is a matrix constructed by data tags. Then it is feasible to bridge the two matrices by using common user vectors or item vectors in rating matrix R and constructed matrix H. At last we can model the algorithm with more information.

3 Building LTMF Algorithm

3.1 Preparing Work

We design an algorithm to generate user cluster and item cluster. The details are shown as Algorithm 1, θ is the threshold that we can decide:

Algorithm 1. Clustering Algorithm

```
Compute rating frequency of user f_u and item f_i
Sort users and items based on f_u and f_i in reverse order
Compute cosine similarity of users and items
t ← 1, U ← φ, I ← φ
for j = 1, 2, ··· do
    if u_j ∉ U then
        U_t ← u_j ∪ u_k | s_{u_j,u_k} > θ_j, u_k ∉ U
        U ← U_t ∪ U, t ← t + 1
    end if
end for
The same as cluster I
```

In this section, we use clusters that have built before to construct local cluster as (user cluster)-(item) preference matrix and (item cluster)-(user) preference matrix by adding the local preferences into the model separately. $X_{u,i}$ and $Y_{u,i}$ can be represented as (user cluster)-(item) and (item cluster)-(user) preference, the details are shown in formulas 5 and 6 :

$$X_{u,i} = \sum_{k \in U} \frac{w_{r_{k,j}} r_{k,j}}{|u|} = (\sum_{k \in U} \frac{w_{p_k} p_k^T}{|U|}) q_j + \omega_U \tag{5}$$

$$Y_{u,i} = \sum_{k \in I} \frac{w_{r_{i,k}} r_{i,kj}}{|I|} = P_i^T (\sum_{k \in I} \frac{w_{q_k} q_k^T}{|I|}) q_j + \omega_I \tag{6}$$

where r is the rating, w is the weight of the corresponding latent factors. $\omega \sim N(0, \sigma^2)$, p and q denote the latent factor of user and item separately.

We use $G_{i,t}$ and $H_{u,t}$ to denote item-tag matrix and user-tag matrix. $G_{i,t}$ is 1 if item i contains tag t, otherwise is 0, same as $H_{u,t}$. Then we construct (user cluster)- (tag) matrix and (item-cluster)-(tag) matrix by using $X_{u,i}$?? and $Y_{u,i}$: The details are shown as formulas 7 and 8:

$$P_{u,t} = \frac{1}{N} \sum X_{u,i} \times G_{i,t} \tag{7}$$

$$Q_{u,t} = \frac{1}{N} \sum Y_{u,i} \times H_{i,t} \tag{8}$$

Now we can combine $R_{u,i}$, $P_{u,t}$ and $Q_{i,t}$ to build LTMF model.

3.2 Building Models

Rating matrix R_{ui} is formed by latent factor matrix U and V. By the definition of $X_{u,i}$ and $Y_{u,i}$, we know that $X_{u,i}$ and $Y_{u,i}$ are the functions of U and V. So

$P_{u,t}$, $Q_{i,t}$ are also the functions of U, V and the tag latent factor T, short for $P_{u,t} = f(U_u^T)T_t$, $Q_{i,t} = g(V_i^T)T_t$. The loss function of LTMF can be organized as formula 9:

$$E = \frac{1}{2}\sum_{u=1}^{M}\sum_{i=1}^{N} I_{ui}^{R}(R_{ui} - U_u^T V_i)^2 + \frac{\alpha}{2}\sum_{u=1}^{M}\sum_{t=1}^{K}(P_{ut} - f(U_u^T)T_t)^2 + \frac{\beta}{2}\sum_{i=1}^{N}\sum_{t=1}^{K} (Q_{i,t} - g(V_i^T)T_t)^2 + \frac{\lambda_U}{2}\sum_{u=1}^{M}||U_u||_F^2 + \frac{\lambda_V}{2}\sum_{i=1}^{N}||V_i||_F^2 + \frac{\lambda_T}{2}\sum_{t=1}^{K}||T_t||_F^2 \tag{9}$$

where $\omega_U = \sigma_R/\sigma_U$, $\omega_V = \sigma_R/\sigma_V$, α and β are the regularization coefficients, K is number of tags, $\lambda_U = \omega_U + \varphi_U$, $\lambda_V = \omega_V + \tau_V$, $\lambda_T = \varphi_T + \tau_T$.

3.3 SGD Training Algorithm

We use SGD algorithm to train the loss function of LTMF model, the detail is shown in Algorithm 2:

Algorithm 2. SGD training algorithm

Input: Rating Matrix R, Cluster Preference Matrix P, Q, Latent Factor Dimension F, Learning Rate η, Scale Factor α and β, Regularization Coefficients λ_U, λ_V, λ_T
Output: U, V

```
Randomly initialize U, V and T with small numbers
while error on validation set decrease do
    ▽_U E = I(U^T V − R)V + αf'(f(U^T)T − P)T + λ_U U
    ▽_V E = [I(U^T V − R)]^T U + βg'(g(V^T)T − Q)T + λ_V V
    ▽_T E = αf(f(U^T)T − P) + βg(g(V^T)T − Q) + λ_T T
    Set η = 0.08
    while E(U − η ▽_U E, V − η ▽_V E, T − η ▽_T E) > E(U, V, T) do
        Set η = η/2
    end while
    U = U − η ▽_U E
    V = V − η ▽_V E
    T = T − η ▽_T E
end while
return U, V
```

4 Experimental Evaluation

4.1 Datasets Description and Compared Models

To evaluate the performance of this algorithm, we use the data provided by http://www.dianping.com/. This datasets contain the information that LTMF algorithm demanded. All the items contain one tag or more, such as district, type of services, subway and same as the user data. This dataset contains 862328 ratings, 28518 items, 127150 users and 47509 tags. Every item or user has one or more tag feature vectors, it equals to 1 if contains the tag otherwise equals to 0.

To demonstrate the performance of LTMF, various methods are studied as baseline. These compared methods are: 1.KNN (memory-based neighbor model algorithm); 2. PMF (model-based matrix factorization algorithm); 3. Tag-LFM (combine tag feature MF algorithm); 4. SBMF (combine neighbor model MF algorithm). This paper uses RMSE to evaluate the performance and applies 5-fold cross validation.

In the experimental process, α and β are used to balance the impact of local structure and tag features on the model. When other parameters are fixed, setting $\alpha = 0.8$ and $\beta = 0.7$ will minimize the RMSE. $\lambda_U = \omega_U + \varphi_U$, $\lambda_V = \omega_V + \tau_V$, $\lambda_T = \varphi_T + \tau_T$, and $\omega_U = \sigma_R/\sigma_U$, $\omega_V = \sigma_R/\sigma_V$. This indicates that λ_U, λ_V and λ_T are compound parameters. But it is practical to set these parameters to a relatively small value,such as $\lambda_U = \lambda_V = \lambda_T = 0.001$.Then we can regulate these values by cross validation in experiments. And the results show that it is acceptable.

4.2 Performance Evaluation

The above Figs. 2 and 3 obviously indicate that with the number of iterations increasing, LTMF performs better than other algorithms.

Figures 4 and 5 show that in a practical environment (means that the number of iterations and the number of observed ratings are relatively more), the model with higher latent factor dimensionality generally performs better.

Figure 6 demonstrates that the algorithms performance is proportional to the number of observed ratings. LTMF combine the existing methods to create

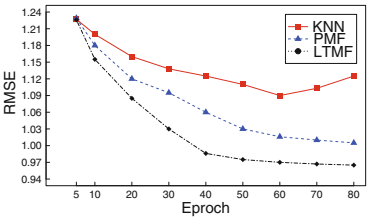

Fig. 2. Impact of number of iterations compared with basic models

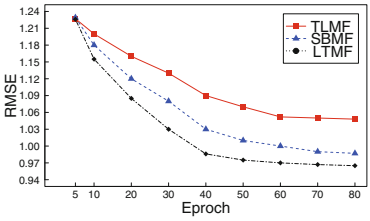

Fig. 3. Impact of number of iterations compared with advanced models

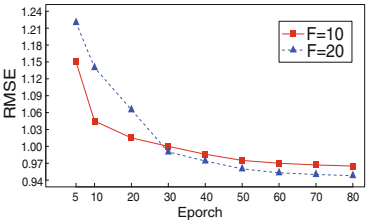

Fig. 4. Impact of latent factor dimensionality with number of iterations

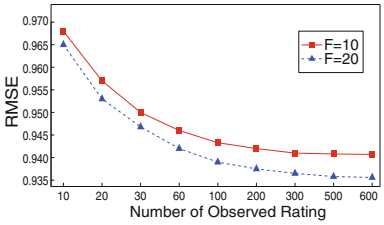

Fig. 5. Impact of latent factor dimensionality with observed ratings

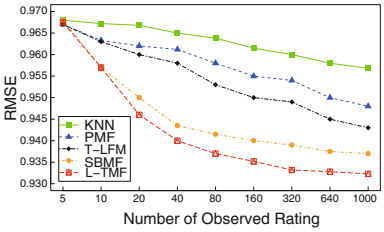

Fig. 6. Impact of observed ratings

a new one. But each stage of the algorithm can be computed separately. So the complexity of LTMF is linear growth, it is quite acceptable when the number of observed ratings increase rapidly.

5 Conclusion

This paper proposes LTMF algorithm and shows that integrating with more information can improve interpretability and performance. But due to constructing new matrices, LTMF can only be applied to the scenarios that contain the information which the algorithm demanded. And also LTMF has more parameters than other traditional models. In the future, we will study on how to tune the parameters more effective and explore other local information such as social networks.

Acknowledgment. This work is supported by National Natural Science Foundation of China (61003031, 61202376), Shanghai Engineering Research Center Project ($GCZX14014$), Shanghai Key Science and Technology Project in IT(14511107902), Shanghai Leading Academic Discipline Project($XTKX2012$) and Hujiang Research Center Special Foundation($C14001$).

References

1. Kim, B.S., Kim, H., Lee, J., Lee, J.-H.: Improving a recommender system by collective matrix factorization with tag information. In: 2014 Joint 7th International Conference on Soft Computing and Intelligent Systems (SCIS), and 15th International Symposium on Advanced Intelligent Systems (ISIS), pp. 980–984. IEEE (2014)
2. Grivolla, J., Badia, T., Campo, D., Sonsona, M., Pulido, J.-M.: A hybrid recommender combining user, item and interaction data. In: 2014 International Conference on Computational Science and Computational Intelligence (CSCI), vol. 1, pp. 297–301. IEEE (2014)
3. Si, L., Jin, R.: Unified filtering by combining collaborative filtering and content-based filtering via mixture model and exponential model. In: Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management, pp. 156–157. ACM (2004)
4. Wang, J., De Vries, A.P., Reinders, M.J.: Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 501–508. ACM (2006)

5. Insuwan, W., Suksawatchon, U., Suksawatchon, J.: Improving missing values imputation in collaborative filtering with user-preference genre and singular value decomposition. In: 2014 6th International Conference on Knowledge and Smart Technology (KST), pp. 87–92. IEEE (2014)
6. Bell, R.M., Koren, Y.: Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In: 2007 Seventh IEEE International Conference on Data Mining ICDM 2007, pp. 43–52. IEEE (2007)
7. Funk, S.: Netflix update: Try this at home, December 2006
8. Mnih, A., Salakhutdinov, R.: Probabilistic matrix factorization. In: Advances in Neural Information Processing Systems, pp. 1257–1264 (2007)
9. Chatti, M.A., Dakova, S., Thus, H., Schroeder, U.: Tag-based collaborative filtering recommendation in personal learning environments. IEEE Trans. Learn. Technol. **6**(4), 337–349 (2013)
10. Pirasteh, P., Jung, J.J., Hwang, D.: Item-based collaborative filtering with attribute correlation: a case study on movie recommendation. In: Nguyen, N.T., Attachoo, B., Trawiński, B., Somboonviwat, K. (eds.) ACIIDS 2014, Part II. LNCS, vol. 8398, pp. 245–252. Springer, Heidelberg (2014)
11. Wang, X., Xu, C.: SBMF: Similarity-based matrix factorization for collaborative recommendation. In: 2014 IEEE 26th International Conference on Tools with Artificial Intelligence (ICTAI), pp. 379–383. IEEE (2014)