ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 2, Issue 4, pp: (67-73), Month: October - December 2014, Available at: www.researchpublish.com

Dealing with Sparse Data: A Dynamic Proposal

[1]Mirza Hameed Baig, [2]Mr.B.Sunil Srinivas, [3]Mr.E.Venkataramana

[1,2,3] Department of Software Engineering, Vidya Vikas Institute of Technology, Ranga Reddy, India

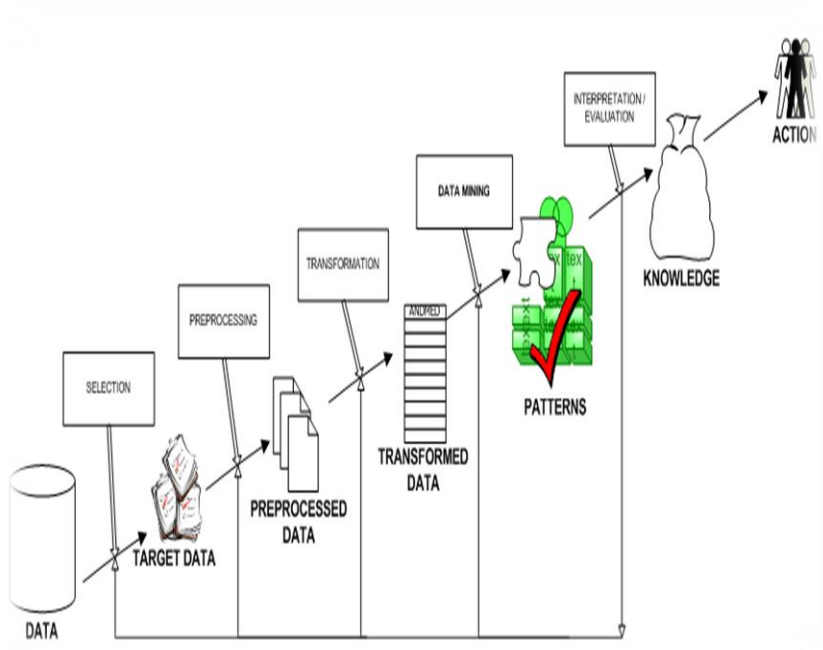
Abstract: **Recommendation techniques are very important in the fields of E-commerce and other Web-based services. One of the main difficulties is dynamically providing high-quality recommendation on sparse data. In this paper, a novel dynamic personalized recommendation algorithm is proposed, in which information contained in both ratings and profile contents are utilized by exploring latent relations between ratings, a set of dynamic features are designed to describe user preferences in multiple phases, and finally a recommendation is made by adaptively weighting the features. Experimental results on public datasets show that the proposed algorithm has satisfying performance.**

Keywords: **E-commerce, Web-based, public data sets.**

I. INTRODUCTION

What is Data Mining?

Generally, data mining (sometimes called data or knowledge discovery) is the process of analyzing data from different perspectives and summarizing it into useful information - information that can be used to increase revenue, cuts costs, or both. Data mining software is one of a number of analytical tools for analyzing data. It allows users to analyze data from many different dimensions or angles, categorize it, and summarize the relationships identified. Technically, data mining is the process of finding correlations or patterns among dozens of fields in large relational databases.



Structure of Data Mining

How Data Mining Works?

While large-scale information technology has been evolving separate transaction and analytical systems, data mining provides the link between the two. Data mining software analyzes relationships and patterns in stored transaction data based on open-ended user queries. Several types of analytical software are available: statistical, machine learning, and neural networks. Generally, any of four types of relationships are sought:

- **Classes**: Stored data is used to locate data in predetermined groups. For example, a restaurant chain could mine customer purchase data to determine when customers visit and what they typically order. This information could be used to increase traffic by having daily specials.
- **Clusters**: Data items are grouped according to logical relationships or consumer preferences. For example, data can be mined to identify market segments or consumer affinities.
- **Associations**: Data can be mined to identify associations. The beer-diaper example is an example of associative mining.
- **Sequential patterns**: Data is mined to anticipate behavior patterns and trends. For example, an outdoor equipment retailer could predict the likelihood of a backpack being purchased based on a consumer's purchase of sleeping bags and hiking shoes.

Data mining consists of five major elements:

1) Extract, transform, and load transaction data onto the data warehouse system.
2) Store and manage the data in a multidimensional database system.
3) Provide data access to business analysts and information technology professionals.
4) Analyze the data by application software.
5) Present the data in a useful format, such as a graph or table.

Characteristics of Data Mining:

- Large quantities of data
- Noisy, incomplete data
- Complex data structure
- Heterogeneous data stored in legacy systems.

Advantages of Data Mining:

1. Marketing / Retail
2. Finance / Banking
3. Manufacturing
4. Governments
5. Law enforcement

II. EXISTING APPROACH

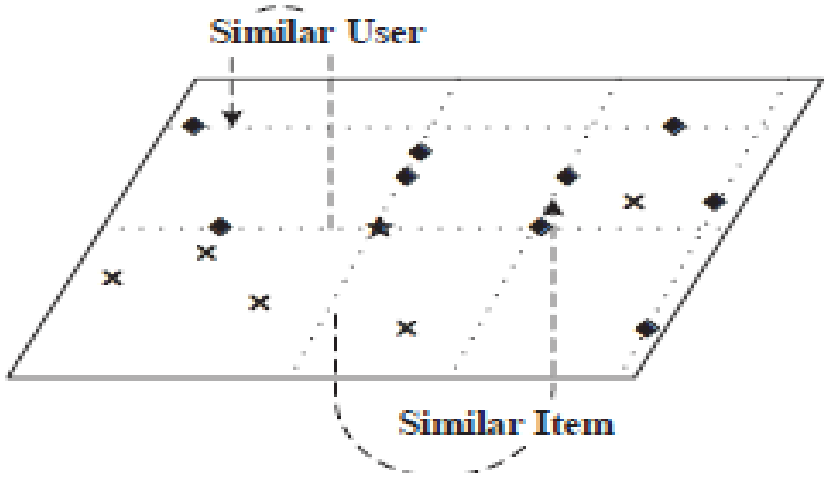
There are mainly three approaches to recommendation engines based on different data analysis methods, i.e., rule-based, content-based and collaborative filtering. Among them, collaborative filtering (CF) requires only data about past user behavior like ratings, and its two main approaches are the neighborhood methods and latent factor models. The neighborhood methods can be user-oriented or item-oriented. They try to find like-minded users or similar items on the basis of co-ratings, and predict based on ratings of the nearest neighbors.

2.1 Drawbacks of Existing Approach

- Proper content cannot be delivered quickly to the appropriate customers.
- No accurate prediction Recommendation.
- Involve most ratings to capture the general taste of users, they still have difficulties in catching up with the drifting signal in dynamic recommendation because of sparsity, and it is hard to physically explain the reason of the involving.

III. PROPOSED APPROACH

In this paper, we present a novel hybrid dynamic recommendation approach. Firstly, in order to utilize more information while keeping data consistency, we use user profile and item content to extend the co-rate relation between ratings through each attribute, as shown in figure.



The main contributions of this paper can be summarized as follows:

(a) More information can be used for recommender systems by investigating the similar relation among related user profile and item content.

(b) A novel set of dynamic features is proposed to describe users' preferences, which is more flexible and convenient to model the impacts of preferences indifferent phases of interest compared with dynamic methods used in previous works, since the features are designed according to periodic characteristics of users' interest and a linear model of the features can catch up with changes in user preferences.

(c) An adaptive weighting algorithm is designed to combine the dynamic features for personalized recommendation, in which time and data density factors are considered to adapt with dynamic recommendation on sparse data.

In most cases, the drifting of users' preferences or items' reputations is not too rapid, which makes it possible to describe temporal state of them by using some features. In this project, firstly we introduce a way to make use of profiles to extend the co-rating relation, and then we propose a set of dynamic features to reflect users' preferences or items' reputations in multiple phases of interest, and after that we propose an adaptive algorithm for dynamic personalized recommendation.

3.1 Relation mining of rating data

For the sparsity of recommendation data, the main difficulty of capturing users' dynamic preferences is the lack of useful information, which may come from three sources - user profiles, item profiles and historical rating records. Traditional algorithms heavily rely on the co-rate relation (to the same item by different users or to different items by the same user), which is rare when the data is sparse. Useful ratings are discovered using the co-rate relation, which is simple, intuitional

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 2, Issue 4, pp: (67-73), Month: October - December 2014, Available at: www.researchpublish.com

and physically significant when we go one or two steps along, but it strongly limits the amount of data used in each prediction.

Instead of searching neighbouring nodes along co-rate edges in the $U \times I$ plane, we try to find a different way to find useful ratings. We notice that when considering the factors which affect a rating $r(u,i)$, we may focus more on some attributes of u and i in their profiles, instead of the user himself or the item itself. For example, if the movie "Gone with the Wind" is given high ratings by middle-aged people and lower ratings by teenagers with no doubt, we would primarily check on the age attribute in a user's profile when predicting probable rating the user would give to the movie, instead of other descriptions of the user or how the user has rated other movies. As is evident, it may not be necessary to stick only to the co-rate relation, and we introduce the *semi-co-rate* relation between ratings whose corresponding user profiles or item contents have similar or identical content in one or more attributes. Since *semi-co-rate* is much less constrained, we extend the co-rate relation to it using user profile and item content, and propose a new way of finding useful ratings for dynamic personalized recommendation.

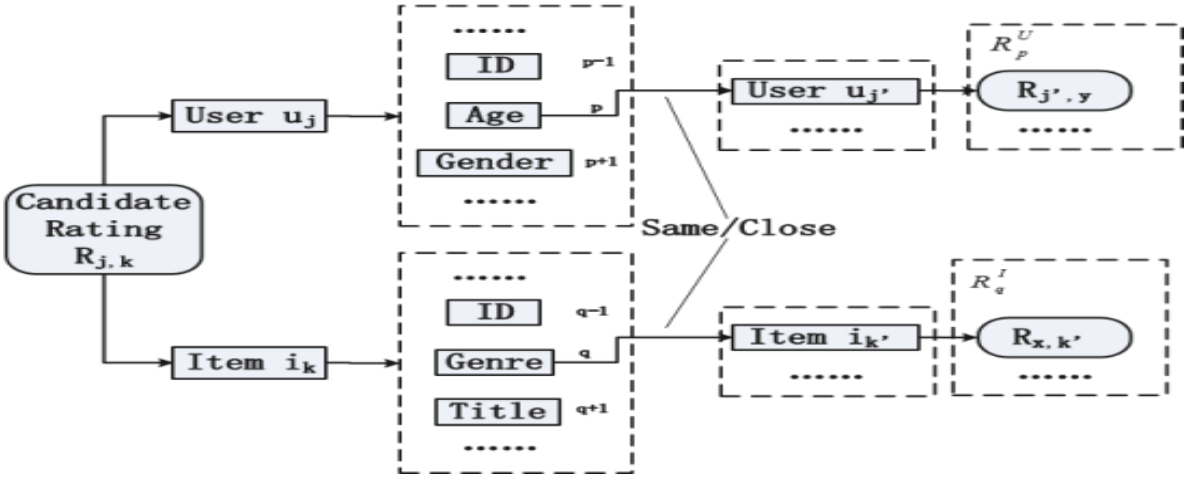


Fig.1 Finding neighboring ratings in the new relation

3.2 Dynamic feature extraction

Users' preferences or items' reputations are drifting, thus we have to deal with the dynamic nature of data to enhance the precision of recommendation algorithms, and recent ratings and remote ratings should have different weights in the prediction. Three kinds of methods were proposed in concept drift to deal with the drifting problem as *instance selection*, *time-window* (usually time decay function) and *ensemble learning*. Koren also proposed an algorithm to isolate transient noise in data using temporal dynamics to help recommendation. These methods help to make progress in precision of dynamic recommendation, but they also have their weaknesses: decay functions cannot precisely describe the evolution of user preferences and only isolating transient noise cannot catch up with the change in data.

So we propose a set of dynamic features to describe users' multi-phase preferences in consideration of computation, flexibility and accuracy. It is impossible to learn weights of all ratings for each user, but it is possible to learn the general weights of ratings in the user's different phases of interest if the phases include ranges of time that are long enough.

In the theory of time series analysis, earlier ratings should impact the predictive features less, and thus they should have lower weights. So if we perform TSA algorithm on a secondary subset of R (i.e. R_s^d) to get a feature $fea_{s,d}$, there would be an uniform formulation as:

$$fea_{s,d} = \sum_{l=1}^{o} \frac{w_l}{w} R_{s,l}^d, \qquad (1)$$

where $\#R_s^d = o$, $R_{s,l}^d (l = 1, 2, ..., o)$ are the rating values which are from the subset R_s^d and listed in reversed time order. And positive weight parameters w_l, (l = 1,2,...,o) and normalization factor w should satisfy

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 2, Issue 4, pp: (67-73), Month: October - December 2014, Available at: www.researchpublish.com

$$\begin{cases} w = \sum_{l=1}^{o} w_l, \\ w_{l_1} \geq w_{l_2} \ if \ l_1 < l_2 \end{cases} \quad (2)$$

Since the subsets are updated frequently, index smoothing [16], which is a classic TSA algorithms, is chosen as the basic TSA algorithm:

$$\begin{cases} R_s^d = \{R_{j',k'} | R_{j',k'} \in R_s \ and \ T_{j,k} - T_{j',k'} \geq T_d\}, \\ fea_{s,d} = \sum_{l=1}^{o} \mu(1-\mu)^{l-1} R_{s,l}^d, \end{cases} \quad (3)$$

where R_s^d (d = 1,2,...) are the secondary subsets, T_d (d = 1,2,...) are a sequence of time differences manually set, $R_{s,l}^d \ (l = 1, 2, ..., o)$ are the rating values listed in reversed order in the subset, μ is the forgetting factor for index smoothing. We have tested different values for mu in the experiments and set μ = 0.95 empirically.

All $fea_{s,d}$ (d = 1,2,...) and the sizes of $R_s^d \ (d = 1, 2, ...)$ are recorded as dynamic features. With the dynamic features, we only have to optimize their weights to get the best estimation of the candidate rating, and in this way we have transformed the training of a recommendation model into weight learning across different secondary rating subsets. Now that the features are related to phases of interest and latent relations between ratings, we would see how the preferences differ with each other in impacting the candidate rating by analyzing optimal weights of the features. We can also see in Eq(3) that the feature extraction does not need heavy computation. Finding all R_s^d needs only comparison in time one by one, and the computation of $fea_{s,d}$ is very efficient. In this way we have proposed a flexible way of feature extraction, where weights in TSA can be different for different rating subsets and the weights for different phases of interest can be variable and learned from the data. The proposed algorithm is termed as *Multiple Phase Division* (MPD).

3.3 Adaptive weighting algorithm

As features like $fea_{s,d}$ (s = 1,2,..., d = 1,2,...) gained by applying *Multiple Phase Division* are all normalized rating values, in other words, as content of user and item profiles have been quantified in the feature extraction, it is convenient for us to organize them for accurate rating estimation by adaptive weighting. Sizes of the relevant subsets are also recorded in MPD and could reflect data density.

We incorporate these features for recommendation with a linear model since they are homogeneous and it is efficient to learn their weights. $\hat{R}_{j,k}$ is used to note the estimated rating that user u_j could give to item i_k at time point $T_{j,k}$, and the adaptive linear model can be formulated as:

$$\hat{R}_{j,k} = \sum_s \sum_d (\alpha_{s,d} + \beta(\#R_s^d)) b_{u_j}(s) b_{i_k}(s) fea_{s,d}, \quad (4)$$

with : $\alpha_{s,d} \geq 0$, $\beta \geq 0$,

where sizes of relevant subsets are used as prior information in weighting the features to improve recommendation accuracy, $fea_{s,d}$ (s = 1,2,..., d = 1,2,...) are the features calculated in Eq.(3), $R_s^d \ (s = 1, 2, ..., d = 1, 2, ...)$ denote their relevant secondary rating subsets, b_{uj} and b_{ik} are binary functions denoting the relating state of candidate rating and relevant subset and $\alpha_{s,d}$ and β are weighting parameters which should balance the weights of features and data density, or, balance the affection of data consistency and quantity of information. In detail, $b_{uj}(s) = 1$ if $R_{j,k}$ is *semi-co-rate* related with all ratings in secondary subset R_s through attribute of the user u_j denoted by s, else $b_{uj}(s) = 0$, $b_{ik}(s) = 1$ if $R_{j,k}$ is *semi-corate* related with all ratings in secondary subset R_s through attribute of the item i_k also denoted by s, else $b_{ik}(s) = 0$.

It is difficult to solve all parameters in Eq.(4) at once, hence we use sequential optimization. Let

$$\delta_{s,d} = \alpha_{s,d} + \beta(\#R_s^d), \quad (5)$$

in Eq.(4) and we first solve for the combined weights $\delta_{s,d}$ (s = 1,2,..., d = 1,2,...) by minimizing the differences between prediction results of the recommendation algorithm and the real rating values in the training set, where RLS algorithm [17] could be used for optimization, i.e.,

$$E = \sum_{R_{j,k} \in R_{Train}} (\hat{R}_{j,k} - R_{j,k}))^2, \quad (6)$$

where R_{Train} is the training set or known rating set. But we notice that a user's preferences or an item's reputations are commonly affected by only a few principle factors, indicating that using more features might also bring noise into the recommendation. So we changed the destination of the optimization and limited the quantity of the features by regularization, and the training problem can be formulated as:

$$\min_{\delta} : \sum_{R_{j,k} \in R_{Train}} (\hat{R}_{j,k} - R_{j,k}))^2 + \lambda ||\delta||_1, \text{ (7) with : } 0 \leq \delta_{s,d} \leq 1 \text{ and } \sum_s \sum_d \delta_{s,d} = 1.$$

This is a typical LASSO optimization problem which can be solved via ADMM [18].

Provided the δs are solved, we turn to the second step of the sequential optimization: to solve αs and β. To deal with the uncertainty in solving αs and β from Eq.(5), we introduce the generalization error like in SVM . Here the generalization error is $\max(\sum_s \sum_d \alpha_{s,d}^2, \ \sum_s \sum_d \beta^2 (\#R_s^d)^2)$, and we minimize it to gain satisfying performance as:

$$\min_{\alpha,\beta} : \max(\sum_s \sum_d \alpha_{s,d}^2, \ \sum_s \sum_d \beta^2 (\#R_s^d)^2), \quad (8) \text{with : } \forall s,d, \ \alpha_{s,d} + \beta(\#R_s^d) = \delta_{s,d}, \ \alpha_{s,d} \geq 0, \ \beta \geq 0.$$

This optimization problem has explicit solution as:

$$\begin{cases} \beta = \dfrac{\sum_s \sum_d \delta_{s,d}}{2 \sum_s \sum_d (\#R_s^d)}, \\ \alpha_{s,d} = \delta_{s,d} - \beta(\#R_s^d) \ \ for \ all \ s,d. \end{cases} \quad (9)$$

Now we have a practical way of solving all the parameters. Firstly we solve δs from Eq.(7) using Lasso algorithm, then use Eq.(8) and Eq.(9) to compute αs and β.

3.4 Advantages of Proposed Approach:

- ✓ Hybrid dynamic recommendation approach.
- ✓ Effective with dynamic data and significantly outperforms previous algorithms.
- ✓ Accurate predication and Recommendation.
- ✓ More information can be used for recommender systems by investigating the similar relation among related user profile and item content.

IV. CONCLUSION

In this paper, we proposed a novel dynamic personalized recommendation algorithm for sparse data, in which more rating data is utilized in one prediction by involving more neighboring ratings through each attribute in user and item profiles. A set of dynamic features are designed to describe the preference information based on TSA technique, and finally a recommendation is made by adaptively weighting the features using information in multiple phases of interest. Experimental results on public MovieLens 100k and Netflix Competition data indicate that the proposed algorithm is effective, and its computational cost is also acceptable.

REFERENCES

[1] B. M. Sarwar, G. Karypis, J. A. Konstan, J. Riedl, Item-based collaborative filtering recommendation algorithms, in: WWW, 2001, pp. 285–295.

[2] P. Brusilovsky, A. Kobsa, W. Nejdl (Eds.), The Adaptive Web, Methods and Strategies of Web Personalization, Lecture Notes in Computer Science, Springer, 2007.

[3] G. Adomavicius, A. Tuzhilin, Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions, IEEE Trans. Knowl. Data Eng. 17 (6) (2005) 734–749.

[4] Y. Koren, Collaborative filtering with temporal dynamics, Communications of the ACM 53 (4) (2010) 89–97.

[5] L. Candillier, F. Meyer, M. Boull´e, Comparing state-of-the-art collaborative filtering systems, in: P. Perner (Ed.), MLDM, Vol. 4571 of Lecture Notes in Computer Science, Springer, 2007, pp. 548–562.

[6] K. Yu, A. Schwaighofer, V. Tresp, X. Xu, H. Kriegel, Probabilistic memory-based collaborative filtering, IEEE Transactions on Knowledge and Data Engineering 16 (1) (2004) 56–69.

[7] F. Fouss, A. Pirotte, J. Renders, M. Saerens, Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation, IEEE TKDE 19 (3) (2007) 355–369.

[8] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer 42 (8) (2009) 30–37.

[9] S. Boutemedjet, D. Ziou, Long-term relevance feedback and feature selection for adaptive content based image suggestion, Pattern Recognition 43 (12) (2010) 3925–3937.

[10] B. M. Kim, Q. Li, C. S. Park, S. G. Kim, J. Y. Kim, A new approach for combining content-based and collaborative filters, J. Intell. Inf. Syst. 27 (1) (2006) 79–91.

[11] G. Prassas, K. C. Pramataris, O. Papaemmanouil, Dynamic recommendations in internet retailing, in: ECIS, 2001.

[12] S. Rendle, C. Freudenthaler, L. Schmidt-Thieme, Factorizing personalized markov chains for next-basket recommendation, in: Proceedings of the 19th WWW, ACM, 2010, pp. 811–820.

[13] C. Xia, X. Jiang, S. Liu, Z. Luo, Z. Yu, Dynamic item-based recommendation algorithm with time decay, in: ICNC, IEEE, 2010, pp. 242–247.

[14] J. Lai, T. Huang, Y. Liaw, A fast k-means clustering algorithm using cluster center displacement, PR 42 (11) (2009) 2551–2556.

[15] Tsymbal, The problem of concept drift: definitions and related work, Computer Science Department, Trinity College Dublin.

[16] X. Tang, C. Yang, J. Zhou, Stock price forecasting by combining news mining and time series analysis, in: Web Intelligence, IEEE, 2009, pp. 279–282.

[17] J. Mohammed, Real-time implementation of an efficient rls algorithm based on iir filter for acoustic echo cancellation, in: IEEE/ACS ICCSA, IEEE, 2008, pp. 489–494.

[18] B. Efron, T. Hastie, I. Johnstone, R. Tibshirani, Least angle regression, The Annals of statistics 32 (2) (2004) 407–499.

[19] B. Boser, I. Guyon, V. Vapnik, A training algorithm for optimal margin classifiers, in: Proceedings of the fifth annual workshop on Computational learning theory, ACM, 1992, pp. 144–152.