Zero-Suppressed BDD and collaborative filtering

Katarina Jovanović[1] and Milena Stanković[2]

Abstract – **Techniques that enable efficient data sets manipulation are very important in recommendation systems. In this paper, we present a method for creating recommendation using collaborative filtering and Zero-suppressed binary decision diagram (Zero-suppressed BDD). The binary decision diagrams (BDDs) are used to analyze problems that occur with large data sets. One of BDD is ZBDD which is suitable for presenting user-product matrix that preserves the history of user's behavior in the collaborative filtering system. Using diagram traversal, we allocate similar users and based on the product which users rated, we create recommendations for a certain user. This way of creating recommendations can be effective with systems that work with large number of users and products.**

Keywords – **Collaborative filtering, Zero-suppressed BDD, Tuple histogram, Pattern histogram.**

I. Introduction

Collaborative filtering system (CF) is one of the ways for the recommendation of products that is based on users' interests and preferences, and selects automatically the products that might be of interest to people. The goal of these systems is to assist in the search and selection of products. The systems for recommendation are based on the assumption that users who have agreed in the choice of products in the past, they will agree in choice of products in the future and they will like a similar kind of products. These users represent a group of similar users.

CF collects feedback from users on the basis of their rates of the products and determines that the users are interested in some products based on their behavior (review, purchase, commenting on the product). History of users' behavior represents the matrix that is called user - product matrix.

Collaborative filtering is widely applied in electronic commerce, where customers can rate and buy different products. On the basis of users' activities, CF generates recommendations that include products of interest. Application of this technique is found in the creation of social networks for recommendation of new friends, groups and pages.

As already known, ZBDD is used to analyze and solve problems that arise in large databases. This data structure manipulates datasets simpler and more efficient than the original BDD (Binary Decision Diagram) and also provides a unique and compact set representation. This paper will present the process of creating recommendations using ZBDD.

[1]Katarina Jovanović is with the Faculty of Electronic Engineering at University of Niš, Aleksandra Medvedeva 14, Niš 1800, Serbia, E-mail: katarina_jovanovic@outlook.com.

[2]Milena Stanković is with the Faculty of Electronic Engineering at University of Niš, Aleksandra Medvedeva 14, Niš 1800, Serbia.

II. Zero-Suppressed BDD

BDDs have been developed to handle Boolean functions, however, they can also be used to represent sets of combinations. The term "sets of combinations" represents a set of elements, where the element is a combination of n items. Examples of such data model are combinations of switching devices (ON/OFF), fault combinations, and sets of paths in the networks [7].

A combination of n items can be represented by an n-bit binary vector, $(x_1 x_2 x_3 \dots x_n)$ where each bit, $x_k \epsilon \{0,1\}$ expresses whether or not the item is included in the combination. ZBDD is a special type of BDDs which manipulates efficiently with combination sets and it is based on the special reduction rules.

- Delete all nodes whose 1-edge directly points to the 0-terminal node, and jump through to the 0-edge's destination, as shown in Fig 1.
- Share equivalent nodes as well as ordinary BDDs.
- Not to delete the nodes whose two edges point to the same node, which used to be deleted by the original rule [7].

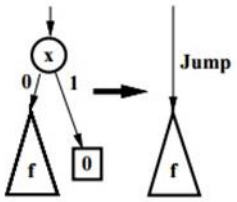


Fig. 1. ZBDD reduction rule

The features of ZBDD.

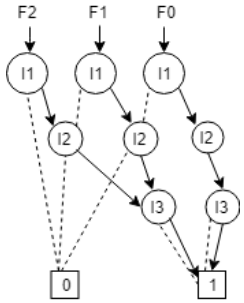
- In ZBDD, the nodes of irrelevant items (those that are not part of any combination) are automatically deleted by ZBDD reduction rules.
- ZBDD is especially effective for representing sparse combinations
- Each path from the root node to 1-terminal node corresponds to each combination in the set. In fact, the number of such paths in ZBDD corresponds to the number of combinations in the set.
- ZBDD structure explicitly stores all items in all combinations, as well as using an explicit linear linked list data structure. [7].

Tuple or transaction is a record that contains a combination of items. The table that contains the number of appearance of each tuple in the given database is tuple histogram.

In the example of Fig. 2 we show a method of representing tuple histogram by using ZBDD. The numbers of tuple's

appearances we decompose into n-digits of ZBDD vector $\{F_0, F_1, \ldots, F_{n-1}\}$, actually we encode the appearance numbers into binary digital code as shown in Fig. 2. F_0 represents a set of tuples appearing odd times, least significant bit, while F_1 represents set of tuples whose second lowest bit of number of appearances is 1. In similar way we define the set of each digit up to F_{n-1}. In the example the tuple frequencies are decomposed as: $F_0 = \{I_1I_2, I_1I_2I_3, I_2I_3\}$, $F_1 = \{I_1I_2, I_2\}$, $F_2 = \{I_1I_2I_3\}$ and each digit can be represented by ZBDD vector. ZBDD vector in Fig. 2 is constructed base on presented tuple-histogram. Also, ZBDDs can share their sub-graphs with each other.

tuple	frequency	F_2	F_1	F_0
I_1I_2	3 (011)	0	1	1
$I_1I_2I_3$	5 (101)	1	0	1
I_2	2 (010)	0	1	0
I_2I_3	1 (001)	0	0	1



$F_0 = \{I_1I_2, I_1I_2I_3, I_2I_3\}$

$F_1 = \{I_1I_2, I_2\}$

$F_2 = \{I_1I_2I_3\}$

Fig. 2. ZBDD vector for tuple histogram

The procedure of generating tuple-histogram F_T for a given database D is shown with pseudocode [7].

$$\begin{aligned} &F_T = 0 \\ &forall\ T\ \in D\ do \\ &\quad F_T = F_T.add(T) \\ &return\ F_T \end{aligned} \qquad (1)$$

F_T is tuple-histogram for given database D, $F.add(T)$ adds combination (tuple) T in the *ZBDD* vector F.
After creating a histogram, it is easy to single out the tuples that appear more than α times. This procedure will be used for creating ZBDD vector. The number of ZBDD nodes in each digit is bounded by total appearance of items in all tuples.

A subset of items included in the tuple represents a pattern. A pattern-histogram is a table that contains the number of appearance of each pattern in any tuple in the given database. In fact, a tuple of k items includes 2^k patterns.
The procedure of generating pattern-histogram based on ZBDD is given by pseudocode [7].

$$\begin{aligned} &F_p = 0 \\ &forall\ T \in D\ do \\ &\quad P = T \\ &\quad forall\ v \in T\ do \\ &\quad\quad P = P\ \cup P.onset(v) \\ &\quad F_p = F_p.add(P) \\ &return\ F_p \end{aligned} \qquad (2)$$

Procedures of generating tuple- histogram and pattern-histogram have been developed by Shin-ichi Minato and his team.

III. ZBDD FOR COLLABORATIVE FILTERING

Recommender systems, within the web sites, keep the history (behavior) for each user in a user-product matrix. User's history can be written as a combination of products that are rated by the user. How the binary decision diagrams are used to represent the combination, the combination of n products can be represented by n-bit binary vector $(x_1x_2x_3 \ldots x_n)$, where each bit, $x_k \epsilon \{0,1\}$, indicates whether the product is the part of the combination or not.

Table I represents the history of the fifteen users and their interest in the products (user-product matrix). Numbers greater than zero indicates that users rated the products while zero indicates that the products are not rated by users.

TABLE I
USER – PRODUCT MATRIX

Product / User	I_1	I_2	I_3	I_4	I_5	I_6	I_7
U_1	5	0	4	0	3	0	2
U_2	0	2	0	3	0	5	0
U_3	4	5	5	0	0	2	3
U_4	0	2	2	0	2	2	0
U_5	5	0	4	0	0	3	0
U_6	2	0	1	0	0	0	0
U_7	1	2	0	0	0	0	0
U_8	0	0	3	3	3	4	5
U_9	5	5	0	0	4	3	2
U_{10}	0	2	2	0	0	3	3
U_{11}	4	0	4	0	5	0	0
U_{12}	0	0	1	5	0	0	0
U_{13}	0	5	0	0	0	4	1
U_{14}	0	3	3	0	0	4	4
U_{15}	5	0	5	0	0	0	0

If we look at the user U_2, from Table I, it can be seen that the user rated the products I_2, I_4 and I_6, while the user U_5 rated I_1, I_3 and I_6 products.

The user for whom system creates a recommendation is an active user and in our case it is U_{13} user. Before creating recommendations, it is necessary to form a group of similar users that are similar to the active user in terms of interests in products. To find similar users, we take advantage of tuple-histogram and operations that can be performed on them. ZBDD vector for tuple histogram is represented in Table II. Each combination represents one path in the ZBDD diagram. Therefore, finding similar users is based on the tour of diagram and extracts paths that match with the paths that are appropriate to the active user in k or more segments.

TABLE II
ZBDD VECTOR FOR TUPLE HISTOGRAM

Tuple	Frequency	F_1	F_0
$I_1I_3I_5I_7$	1(01)	0	1
$I_2I_4I_6$	1(01)	0	1
$I_1I_2I_3I_6I_7$	1(01)	0	1
$I_2I_3I_5I_6$	1(01)	0	1
$I_1I_3I_6$	1(01)	0	1
I_1I_3	2(10)	1	0
I_1I_2	1(01)	0	1
$I_3I_4I_5I_6I_7$	1(01)	0	1
$I_1I_2I_5I_6I_7$	1(01)	0	1
$I_2I_3I_6I_7$	2(01)	1	0
$I_1I_3I_5$	1(01)	0	1
I_3I_4	1(01)	0	1
$I_2I_6I_7$	1(01)	0	1

$$F_0 = \begin{Bmatrix} I_1I_3I_5I_7, I_2I_4I_6, I_1I_2I_3I_6I_7, I_2I_3I_5I_6, I_1I_3I_6, I_1I_2, \\ I_3I_4I_5I_6I_7, I_1I_2I_5I_6I_7, I_1I_3I_5, I_3I_4, I_2I_6I_7 \end{Bmatrix}$$
$$F_1 = \{I_1I_3, I_2I_3I_6I_7\}$$

Operation, which will be used in determining similar users, is finding all combinations that contain given pattern.

$$\begin{aligned} &S = \cup F_k \\ &forall\ v \in P\ do: \\ &\ S = S.onset(v).change(v) \\ &return\ S \end{aligned} \quad (3)$$

S is union of tuples which contain pattern *P*, *S.onset(v)* selects the subset of combinations including item *v* and *S.change(v)* inverts existence of *v* (add / delete) on each combination (see table with primitive ZBDD operations [7]).

Patterns with one element that match the active user's combination is $P \rightarrow \{I_2, I_6, I_7\}$. *ZBDDsimilarUsers* algoritham finds all combinations which contain patterns from the set *P*. Combinations that contain α or more patterns from set *P* belong to group of similar users. The *ZBDDsimilarUsers* algorithm is shown in Fig 3.

```
ZBDDsimilarUsers(F)
{
    S = ∪ F_k
    forall v ∈ P do:
      S = S.onset(v).change(v)
    forall T ∈ S
      countT = 0
      for i = 1 to S.count
        if(T = S[i])
          countT = countT + 1
      if(countT ≥ α)
        S_u = S_u U T
    return S_u
}
```

Fig. 3. Find similar users algorithm

All combinations are stored in a collection *S*. The parameter α depends on the recommendation system and represents number that indicates that two users are similar if they rated α or more the same products.

If the combination *T* from set *S* appears α or more times, it means that the active user and the user to whom refers this combinations are similar. All combinations that satisfy the requirement are placed in a collection S_u which contains all similar users.

In order to determine the product group for the recommendation, it is necessary for each combination from the set S_u to extract patterns that contain one element. If the pattern is part of active user's combinations, it will not be included in the group for recommendation while other patterns will be included in recommendation. A set of products for a recommendation is obtained by separating nodes that are not the part of active user's path in ZBDD. If *Q* represents patterns of similar users, *P* represents patterns of the active users and *R* represents products for recommendation, procedure to extract the products is:

$$R = Q - Q \cap P \quad (4)$$

After determining group of products for recommendation it is needed to define top *N* products for recommendation and order of recommendation. By computing the average rate of products from the group and sorting them by the average rates, we obtain the order of the products for the recommendation. The average product's rate represents weight of the branch in ZBDD vector.

IV. CONCLUSION

In this paper we presented a method for creating recommendations using collaborative filtering and ZBDD. This way of creating recommendation provides simpler manipulation with data sets. Here are given procedures for finding similar users and creating group of products for recommendation. Finding similar users is based on a touring of ZBDD diagram and extracting sub-diagram which represents similar users. The nodes in the ZBDD sub-diagram, which are not part of the active user's path, are products for recommendation. Sorting the products by rates we determine the order in which products will be recommended to active user.

REFERENCES

[1] D. Asanov, "Algorithms and Methods in Recommender System", Berlin Institute of Technology, Berlin, Germany.
[2] M. D. Ekstran, J. T. Riedl. and J. A. Konstan, "Collaborative Filtering Recommender Systems", Foundations and Trends® in Human–Computer Interaction: vol. 4, no. 2, pp. 81–173, 2011.

[3] M. Jones, "*Recommender systems"*, http://www.ibm.com/developerworks/library/os-recommender1/, accessed on 01.05.2017.

[4] J. Lee., M. Sun, G. Lebanon, "A Comparative Study of Collaborative Filtering Algorithm", Workshops, Demos, and ArXiv Preprints, 2012

[5] P. Melville and V. Sindhwani, "Recommender system", In: C. Sammut and G. Webb, Eds., Encyclopedia of Machine Learning, Springer, Berlin, 2010, pp. 829-838.

[6] S. Minato and H. Arimura , "Generating Frequent Closed Item Sets Based on Zero-suppressed BDDs", Hokkaido University, Division of Computer Science, TCS Technical Reports, TCS-TR-A-06-17, Jul. 2006

[7] S. Minato and H. Arimura, "Combinatorial Item Set Analysis Based on Zero-Suppressed BDDs", Hokkaido University, Division of Computer Science, TCS Technical Reports, TCS-TR-A-04-1, Dec, 2004.

[8] X. Su and T. M. Khoshgoftaar, "A Survey of Collaborative Filtering Techniques", *Advances in Artificial Intelligence*, vol. 2009, Article ID 421425, 19 pages, 2009.

[9] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, "Item-based collaborative filtering recommendation algorithm", In Proceedings of the 10th International Conference on World Wide Web (pp. 285-295). (WWW '01). New York, NY, USA: ACM.