Incremental Collaborative Filtering for Highly-Scalable Recommendation Algorithms

Manos Papagelis[1, 2], Ioannis Rousidis[2], Dimitris Plexousakis[1, 2], and Elias Theoharopoulos[3,*]

[1] Institute of Computer Science, Forth, Heraklion, Greece
{papaggel, dp}@ics.forth.gr
[2] Computer Science Department, University of Crete, Heraklion, Greece
rousidis@csd.uoc.gr
[3] School of Informatics, University of Edinburgh, Edinburgh, Scotland
e.theoharopoulos@sms.ed.ac.uk

Abstract. Most recommendation systems employ variations of Collaborative Filtering (CF) for formulating suggestions of items relevant to users' interests. However, CF requires expensive computations that grow polynomially with the number of users and items in the database. Methods proposed for handling this scalability problem and speeding up recommendation formulation are based on approximation mechanisms and, even when performance improves, they most of the time result in accuracy degradation. We propose a method for addressing the scalability problem based on incremental updates of user-to-user similarities. Our Incremental Collaborative Filtering (ICF) algorithm (i) is not based on any approximation method and gives the potential for high-quality recommendation formulation (ii) provides recommendations orders of magnitude faster than classic CF and thus, is suitable for online application.

1 Introduction

Recommendation algorithms are extensively adopted by both research and e-commerce applications, in order to provide an intelligent mechanism to filter out the excess of information available in a domain [1]. Collaborative filtering (CF) [3], almost certainly, is the key method to effortlessly find out items that users will probably like according to their logged history of prior transactions.

However, CF requires computations that are very expensive and grow polynomially with the number of users and items in a system. Therefore, in order to bring recommendation algorithms effectively on the web, and succeed in providing recommendations with high accuracy and acceptable performance, sophisticated data structures and advanced, scalable architectures are required [7]. To address this scalability problem, we present an incremental CF method, based on incremental updates of user-to-user similarities which is also able to recommend items orders of magnitude faster than classic CF, while maintaining the recommendation quality.

[*] Work conducted while at ICS-FORTH.

M.-S. Hacid et al. (Eds.): ISMIS 2005, LNAI 3488, pp. 553 – 561, 2005.

The remainder of the paper is organized as follows: Section 2 elaborates on the scalability challenge and explains the weaknesses of already proposed methods. Section 3 presents our Incremental CF method. Section 4 argues about complexity issues of the algorithms, while Section 5 presents our experimental evaluation. Section 6 concludes our work and discusses further research directions.

2 The Scalability Challenge for Collaborative Filtering

Classic CF algorithm generates recommendations based on a subset of users that are most similar to the active user. The formulation of a single recommendation is a two-step computation. First, the algorithm needs to compute the similarity between the active user and all other users, based on their co-rated items, so as to pick the ones with similar behavior. Subsequently, the algorithm recommends to the active user items that are highly rated by his or her most similar users. In order to compute the similarities between users, a variety of similarity measures have been proposed, such as Pearson correlation, cosine vector similarity, Spearman correlation, entropy-based uncertainty measure and mean-square difference. However, Breese et al. [4] and Herlocker et Al. [5] suggest that Pearson correlation performs better than all the rest.

If we define the user-item matrix as the matrix having as elements the ratings of users to items, then a user's model is represented in this matrix as an *n*-dimensional vector, where *n* is the number of items in the database. This vector is extremely sparse for most users, since, even ones that are very active result in rating just a few of the total number of items available in a database. If we define the subset of items that users u_x and u_y have co-rated as $I'=\{i_x: x=1, 2, \ldots, n' \text{ and } n' \leq n\}$, where *n* is the total number of items in the database, r_{u_x,i_h} as the rating of user u_x to item i_h and $\overline{r_{u_x}}$, $\overline{r_{u_y}}$ as the average ratings of users u_x and u_y respectively, then the similarity between two users is defined as the Pearson correlation of their associated rows in the user-item matrix and is given by equation 1 [14].

	i_1	…	i_x	…	i_y	…	i_n
…							
u_x	$r_{x,1}$		$r_{x,x}$		$r_{x,y}$		$r_{x,n}$
…							
u_y	$r_{y,1}$		$r_{y,x}$		$r_{y,y}$		-
…							

$$sim(u_x, u_y) = \frac{\sum_{h=1}^{n'} (r_{u_x,i_h} - \overline{r_{u_x}})(r_{u_y,i_h} - \overline{r_{u_y}})}{\sqrt{\sum_{h=1}^{n'} (r_{u_x,i_h} - \overline{r_{u_x}})^2} \sqrt{\sum_{h=1}^{n'} (r_{u_y,i_h} - \overline{r_{u_y}})^2}} \quad (1)$$

Classic CF fails to scale up its computation with the growth of both the number of users and items in the database. To deal with the scalability problem Breese et al [4] and Ungar et al [8] utilize Bayesian network and clustering approaches, while Sarwar et al [6, 11] apply folding in Singular Value Decomposition (SVD) to reduce the dimensionality of the user-item matrix. It is also possible to address these scaling issues by data reduction or data focusing techniques. Yu et al [12] and Zeng et al [9] adopt instance selection for removing the irrelevant and redundant instances. Moreover, content-boosted CF approaches reduce the number of items examined, by partitioning the item space according to item category or subject classification. Finally, more greedy approaches concentrate on randomly sampling users, discarding users with few ratings or discarding very popular or unpopular items.

Unfortunately, even when these methods achieve improved performance, they also reduce recommendation quality in several ways. Bayesian networks may prove practical for environments in which user preferences change slowly with respect to the time needed to build the model, but are not suitable for environments in which user preference models must be updated frequently. Clustering-based methods suffer from poor accuracy. It is possible to improve their quality by using numerous fine-grained segments [13], but then online user segment classification becomes almost as expensive as finding similar users using the classic CF. SVD-based work focuses mainly on accuracy rather than efficiency. Data focusing and reduction approaches, such as instance selection or item-space partitioning, experience reduced accuracy due to loss of information. If an algorithm discards the most popular or unpopular items, there may be items that will never be recommended to some users. Obviously, to gain in computation one needs to lose in recommendation quality and vice versa. Appropriate trade-offs must be considered.

3 Incremental Collaborative Filtering

In this section, we present a method to deal with the scalability challenge of Classic CF without compromising its recommendation quality. We refer to this method as Incremental Collaborative Filtering (ICF), because it is based on incremental updates of the user-to-user similarities. ICF can be employed to effectively bring highly scalable and accurate recommendation algorithms on the Web.

Whenever a user u_x submits a new rating or updates the value of an already submitted rating, his or her similarity value with the rest of the users may need to be re-computed. Our objective is to express the new similarity value between two users in relation to their old similarity value. This describes an incremental update of their associated similarity. To smoothen the progress of this task we adopt the following notation for the Pearson Correlation similarity measure of equation 1:

$$A = \frac{B}{\sqrt{C}\sqrt{D}} \Rightarrow A = sim(u_x, u_y), \quad B = \sum_{h=1}^{n'} (r_{u_x,i_h} - \overline{r_{u_x}})(r_{u_y,i_h} - \overline{r_{u_y}}), \quad C = \sum_{i=1}^{n'} (r_{u_x,i_h} - \overline{r_{u_x}})^2, \quad D = \sum_{i=1}^{n'} (r_{u_y,i_h} - \overline{r_{u_y}})^2$$

Actually, we split the similarity measure into three factors B, C, D, independently calculate the new values of each factor B', C', D' and then combine these values so as to obtain the value of the new similarity A' as shown below:

$$A' = \frac{B'}{\sqrt{C'}\sqrt{D'}} \Rightarrow A' = \frac{B+e}{\sqrt{C+f}\sqrt{D+g}}, \quad B' = B+e, \quad C' = C+f, \quad D' = D+g$$

where e, f, g are *increments* that need to be computed after either the submission of a new or the update of an existing rating. Next, we split our study, so as to consider the different computations needed for the two special cases. Table 1 shows the increments to be computed and the Appendix provides proof of equations 2-13.

Submission of a new rating: To calculate the similarity of u_a and u_y, when the active user u_a submits a *new rating* for the active item i_a, we distinguish between two cases:

i. u_y had rated i_a: B, C, D are updated due to the new average of u_a, the new rating of u_a to i_a and the new number of co-rated items
ii. u_y had not rated i_a: B, C are updated due to the new average of u_a.

Table 1. Summary of the increments that need to be calculated

		Submission of a new rating	Update of an existing rating
u_y had rated i_a	e	$e = (r_{u_a,i_a} - \overline{r_{u_a}}')(r_{u_y,i_a} - \overline{r_{u_y}}) - \sum_{h=1}^{n'} d\overline{r_{u_a}}(r_{u_y,i_h} - \overline{r_{u_y}})$ **(2)**	$e = dr_{u_a,i_a}(r_{u_y,i_a} - \overline{r_{u_y}}) - \sum_{h=1}^{n'} d\overline{r_{u_a}}(r_{u_y,i_h} - \overline{r_{u_y}})$ **(8)**
	f	$f = (r_{u_a,i_a} - \overline{r_{u_a}}')^2 + \sum_{h=1}^{n'} d\overline{r_{u_a}}^2 - 2\sum_{h=1}^{n'} d\overline{r_{u_a}}(r_{u_a,i_h} - \overline{r_{u_a}})$ **(3)**	$f = dr_{u_a,i_a}{}^2 + 2dr_{u_a,i_a}(r_{u_a,i_a} - \overline{r_{u_a}}') + \sum_{h=1}^{n'} d\overline{r_{u_a}}'^2 - 2\sum_{h=1}^{n'} d\overline{r_{u_a}}'(r_{u_a,i_h} - \overline{r_{u_a}})$ **(9)**
	g	$g = (r_{u_y,i_a} - \overline{r_{u_y}})^2$ **(4)**	$g = 0$ **(10)**
u_y had not rated i_a	e	$e = -\sum_{h=1}^{n'} d\overline{r_{u_a}}(r_{u_y,i_h} - \overline{r_{u_y}})$ **(5)**	$e = -\sum_{h=1}^{n'} d\overline{r_{u_a}}(r_{u_y,i_h} - \overline{r_{u_y}})$ **(11)**
	f	$f = \sum_{h=1}^{n'} d\overline{r_{u_a}}^2 - 2\sum_{h=1}^{n'} d\overline{r_{u_a}}(r_{u_a,i_h} - \overline{r_{u_a}})$ **(6)**	$f = \sum_{h=1}^{n'} d\overline{r_{u_a}}'^2 - 2\sum_{h=1}^{n'} d\overline{r_{u_a}}'(r_{u_a,i_h} - \overline{r_{u_a}})$ **(12)**
	g	$g = 0$ **(7)**	$g = 0$ **(13)**

Table 2. Computation of factors that appear in increments e, f and g

Factors	Calculation
B, C, D	Cached Information (For all pairs of users)
q	Cached Information (The number of the items that a user has rated)
$\overline{r_{u_a}}$, $\overline{r_{u_y}}$	Cached information (Average ratings of all users in database)
$\sum_{h=1}^{n'} r_{u_y,i_h}$, $\sum_{h=1}^{n'} r_{u_a,i_h}$	Cached Information (For each pair of users, the sum of their ratings to co-rated items is cached)
$\overline{r_{u_a}}'$	New average rating of active user: • Submission of a new rating: $\overline{r_{u_a}}' = \frac{r_{u_a,i_a}}{q+1} + \frac{q}{q+1}\overline{r_{u_a}}$ • Update of existing rating: $\overline{r_{u_a}}' = \frac{dr_{u_a,i_h}}{q} + \overline{r_{u_a}}$
r_{u_a,i_a}	Interface (Actual rating of the active user u_a to the active item i_a)
$d\overline{r_{u_a}}$	$d\overline{r_{u_a}} = \overline{r_{u_a}}' - \overline{r_{u_a}} \Leftrightarrow \overline{r_{u_a}}' = \overline{r_{u_a}} + d\overline{r_{u_a}}$ (The difference of user's previous and current average rating)
r_{u_y,i_a}	Database query. (The rating of the user u_y to the item i_a)

Update of an existing rating: To calculate the similarity of u_a and u_y, when the active user u_a updates a rating for the active item i_a, we distinguish between two cases:

i. u_y had rated i_a: B, C are updated due to the new average of u_a and the new rating of u_a to i_a
ii. u_y had not rated i_a: B, C are updated due to the new average of u_a

We expressed B', C' and D' using the former values of B, C and D and the respective increments e, f, g. However, to enable the incremental computation of the new similarities with trivial operations we need to employ an appropriate data structure. We define a caching scheme, in terms of tables in a database, which permits to store the values of B, C and D for all pairs of users. Furthermore, we cache the average rating and the number of items that each user has rated. Cached information needs to be updated after the submission of a new or the update of an existing rating. Table 2 explains how each factor of e, f, g increments is computed.

4 Complexity Issues

In this section, we discuss the computational complexity of the classic CF and ICF algorithms. We initially present worst cases and then try to give approximations of the algorithms under real conditions. For each case our study spans in two directions: the one refers to the complexity of maintaining the user similarities matrix and the other refers to the complexity of formulating one recommendation to an active user.

In case of Classic CF, if m is the number of users and n the number of items in the database, then the computation complexity of maintaining the user similarities matrix is $O(m^2n)$, since we need to compute the similarity between each pair of users according to the subset of their co-rated items. In order to deal with this task, major e-commerce systems prefer to carry out expensive computations offline and feed the database with updated information periodically [2]. In this way, they succeed in providing quick recommendations to users, based on pre-computed similarities. These recommendations however, are not of the highest accuracy as ratings submitted between two offline computations are not considered. Thus, the offline computation method may be detrimental to new or obscure users and items due to their almost undeveloped profile. Alternatively, if user similarities are not pre-computed offline, they may be computed at the time a recommendation is requested. In this case, instead of computing the whole user similarities matrix, we need to only compute the similarities between the active user and all the rest or a set of training users. The cost of this computation is $O(mn)$. The cost of generating a recommendation using the Classic CF is the cost of finding the most similar users to the active user and scanning their rated items to find the ones that are highly rated. This computation costs $O(n)$ when similarities are pre-computed offline and $O(mn)$ otherwise.

In the case of the ICF algorithm, user-to-user similarities are computed incrementally at the time of rating activity and not at the time that a recommendation is requested. The complexity of this operation is $O(mn)$, as at most $m-1$ similarities need to be updated and at most n items need to be examined for each user. Since user similarities are considered pre-computed, the cost of generating a recommendation using ICF is $O(n)$, as n items need to be examined in the worst case.

Due to the high sparsity levels in recommendation systems, it is essential to also consider approximations of the complexities under real conditions. Thus, we define: *m'*, with $m' << m$, the number of users with whom the active user has at least one co-rated item; *n'*, with $n' << n$, the number of items that have not been rated by the active user and have been rated by at least one of its similar users; *n''*, with $n'' << n$, the number of co-rated items of the active user and another user.

According to these definitions, we can set up the approximations of the complexities following the discussion of the previous paragraph. Worst case and approximation complexities of Classic CF and ICF are summarized in Table 3.

Table 3. Worst case and approximation complexities of Classic CF and ICF

<table>
<tr><th></th><th colspan="2">Classic CF</th><th colspan="2">Incremental CF</th></tr>
<tr><th></th><th>Worst</th><th>Approximation</th><th>Worst</th><th>Approximation</th></tr>
<tr><td>Complexity for maintaining the Similarity Matrix</td><td>$O(m^2n)$</td><td>$O(mm'n'')$</td><td>$O(mn)$</td><td>$O(m'n')$</td></tr>
<tr><td rowspan="3">Complexity for providing a recommendation to active user</td><td>$O(mn)$</td><td>$O(m'n'') + O(n')$</td><td rowspan="3">$O(n)$</td><td rowspan="3">$O(n')$</td></tr>
<tr><td colspan="2">Pre-computed Offline</td></tr>
<tr><td>$O(n)$</td><td>$O(n')$</td></tr>
</table>

5 Experimental Evaluation

As complexity computation fails to give real-time performance and behavior of the algorithms described, we set up an experimental scenario for evaluating the performance of our ICF algorithm as opposed to the Classic CF. The evaluation is carried out according to *response time* and *accuracy* metrics as defined below:

Response time: The time required by the algorithm to formulate a recommendation.

Accuracy: The percentage of items an algorithm recommends from the set of items that are recommended by the Classic CF that considers the whole dataset available.

The assumption made here is that recommendations based on the whole dataset are of the highest quality, which is not necessarily true. Indeed, we define this to demonstrate the potential that ICF gives for formulating recommendations based on the complete information in a database and not only a part of it.

The experimental scenario is set up so as to depict the level of scalability that both algorithms demonstrate when the active user requests a single recommendation. We employ sparsity level of 92% and consider a user to be similar to the active user if their associated Pearson correlation coefficient is greater than 0.65 (in a range of -1 to 1). The values selected represent typical values for recommendation systems and do not influence the results of the experiments. Table 4 presents the results of our experiments for user-item matrix of size 100x100 and 1000x1000 respectively. Experiments have been carried out on a 2.80 Mhz, 1G RAM PC.

Table 4. Response time and accuracy performance of Classic CF and ICF

User-item matrix size	Classic CF (Based on sampling)			Incremental CF	
	Samples (#users)	Time (sec)	Accuracy	Time (sec)	Accuracy
100 users x 100 items	10	0.17	22%	0.045	100%
	30	0.55	49.5%		
	50	0.765	67.5%		
	99	1.38	100%		
1000 users x 1000 items	100	6.81	26,7%	0.46	100%
	300	20	53,8%		
	500	33	66.8%		
	999	66	100%		

The following remarks can be drawn from Table 4 about the performance of CF and ICF:

Remark 1: The trade-off between performance and accuracy in case of Classic CF is confirmed. Indeed, Classic CF is very sensitive to the size of samples used. As the sample size increases, accuracy is improved, but the response time also increases and vice versa. Large sample sizes are impractical for online applications due to the slow response time, while small sample sizes are impractical due to accuracy degradation. On the other side, the accuracy of ICF remains as high as 100%, since it is applied to the whole information available.

Remark 2: ICF proves to be highly scalable, as its response time remains acceptable even for a very large data set. For instance, it provides a recommendation in 0.46 seconds for a matrix size of 1000x1000. Classic CF requires extremely disproportional time to reach a satisfactory accuracy level for large matrix sizes. For instance, when an accuracy level of 66.8% is intended, using a sample of 500 users, in a 1000x1000 matrix Classic CF performs 71 times slower than ICF.

Remark 3: The performance of ICF grows linearly with the number of items, thus in case of a *very* large matrix an approximation method may need to be employed.

6 Conclusions and Future Work

High dimensionality seems to be the "Achilles' heel" for most of the CF-based recommendation systems. For dealing with this scalability problem, we proposed an incremental method that replaces expensive vector operations with a scalar operation, able to speed-up computations of high dimensional user-item matrices. We named this method Incremental Collaborative Filtering (ICF). ICF is not based on any approximation method and thus, provides the potential of formulating high-quality recommendations. Moreover, pre-computed user-to-user similarities allow recommendations to be delivered orders of times faster than with classic CF. ICF appears to be suitable for online applications, while the methodology described is general and may be easily adopted to scale the application of CF in other areas. As future directions of our research we consider the development of a method for alleviating the *sparsity* problem of CF. The main idea of the method is to infer trust between users through profile similarities for constructing a denser user-item matrix.

References

1. Sarwar, B., Karypis, G., Konstan, J., Riedl J.: Analysis of recommendation algorithms for e-commerce. Proc. of ACM Electronic Commerce (2000)
2. Linden, G., Smith, B., York, J.: Amazon.com Recommendations: Item-to-Item Collaborative Filtering. IEEE Internet Computing, January 2003
3. Herlocker, J. L., Konstan, J. A., Riedl, J.: Explaining Collaborative Filtering Recommendations. Proc. of the ACM Conf.on CSCW (2000)
4. Breese, J. S., Heckerman, D., Kadie, C.: Empirical analysis of predictive algorithms for collaborative filtering. Proc. of the UAI (1998)
5. Herlocker, J. L., Konstan, J. A., Borchers, A., Riedl, J.: An Algorithmic Framework for Performing Collaborative Filtering. Proc. of ACM SIGIR (1999)
6. Sarwar, B. M., Karypis, G., Konstan, J. A., Riedl, J. T.: Application of Dimensionality Reduction in Recommender System: A Case Study. Proc. of ACM SIGKDD (2000)
7. Papagelis M, Plexousakis D.: Qualitative Analysis of User-based and Item-based Prediction Algorithms for Recommendation Agents. Proc. of CIA (2004)
8. Ungar, L., Foster, D.: Clustering Methods for Collaborative Filtering. Proc. of Workshop on Recommendation Systems, AAAI Press (1998)
9. Zeng, C., Xing, C., Zhou, L.: Similarity Measure and Instance Selection for Collaborative Filtering. Proc. of WWW (2003)
10. Sarwar, B.M., Karypis, G., Konstan, J., Riedl, J.: Incremental SVD-Based Algorithms for Highly Scaleable Recommender Systems. Proc.of ICCIT (2002).
11. Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., Harshman, R.: Indexing by Latent Semantic Analysis. JASIS 41(6) (1990)
12. Yu, K., Xu, X., Tao, J., Ester, M., Kriegel H.: Instance Selection Techniques for Memory-Based Collaborative Filtering. Proc. of SDM (2002).
13. Jung, S. Y., Kim, T.: An Incremental Similarity Computation Method in Agglomerative Hierarchical Clustering. Proc. of ISAIS (2001)
14. Pearson K.: Mathematical contribution to the theory of evolution: VII, on the correlation of characters not quantitatively measurable. Phil. Tr. R. Soc. Lond. A, 195, 1-47, 1900

Appendix: Proof of Equations 2-13

Proof of Equation 2

$$B' = \sum_{h=1}^{n^*} (r_{u_a,i_h} - \overline{r_{u_a}}')(r_{u_y,i_h} - \overline{r_{u_y}}) \Leftrightarrow B' = (r_{u_a,i_a} - \overline{r_{u_a}}')(r_{u_y,i_a} - \overline{r_{u_y}}) + \sum_{h=1}^{n'} (r_{u_a,i_h} - \overline{r_{u_a}}')(r_{u_y,i_h} - \overline{r_{u_y}}) \Leftrightarrow$$

$$B' = (r_{u_a,i_a} - \overline{r_{u_a}}')(r_{u_y,i_a} - \overline{r_{u_y}}) + B - \sum_{h=1}^{n'} d\overline{r_{u_a}}(r_{u_y,i_h} - \overline{r_{u_y}}) \Rightarrow e = (r_{u_a,i_a} - \overline{r_{u_a}}')(r_{u_y,i_a} - \overline{r_{u_y}}) - \sum_{h=1}^{n'} d\overline{r_{u_a}}(r_{u_y,i_h} - \overline{r_{u_y}})$$

Proof of Equation 3

$$C' = \sum_{h=1}^{n^*} (r_{u_a,i_h} - \overline{r_{u_a}}')^2 \Leftrightarrow C' = (r_{u_a,i_a} - \overline{r_{u_a}}')^2 + \sum_{h=1}^{n'} (r_{u_a,i_h} - \overline{r_{u_a}}')^2 \Leftrightarrow$$

$$C' = (r_{u_a,i_a} - \overline{r_{u_a}}')^2 + C + \sum_{h=1}^{n'} d\overline{r_{u_a}}^2 - 2\sum_{h=1}^{n'} d\overline{r_{u_a}}(r_{u_a,i_h} - \overline{r_{u_a}}) \Rightarrow f = (r_{u_a,i_a} - \overline{r_{u_a}}')^2 + \sum_{h=1}^{n'} d\overline{r_{u_a}}^2 - 2\sum_{h=1}^{n'} d\overline{r_{u_a}}(r_{u_a,i_h} - \overline{r_{u_a}})$$

Proof of Equation 4

$$D' = \sum_{h=1}^{n^*} (r_{u_y,i_h} - \overline{r_{u_y}})^2 \Leftrightarrow D' = (r_{u_y,i_a} - \overline{r_{u_y}})^2 + \sum_{h=1}^{n'} (r_{u_y,i_h} - \overline{r_{u_y}})^2 \Leftrightarrow D' = (r_{u_y,i_a} - \overline{r_{u_y}})^2 + D \Rightarrow g = (r_{u_y,i_a} - \overline{r_{u_y}})^2$$

Proof of Equation 5, 6, 7

In the case that user u_y has not rated the item i_a, the values of *B*, *C* and *D* are proved in way similar to equations 2, 3 and 4 respectively. In this case the increments e, f and g equal to:

$$e = -\sum_{h=1}^{n'} d\overline{r_{u_a}}(r_{u_y,i_h} - \overline{r_{u_y}}), \quad f = \sum_{h=1}^{n'} d\overline{r_{u_a}}^2 - 2\sum_{h=1}^{n'} d\overline{r_{u_a}}(r_{u_a,i_h} - \overline{r_{u_a}}), \quad g = 0$$

Proof of Equation 8

$$B' = \sum_{h=1}^{n'} (r_{u_a,i_h} - \overline{r_{u_a}}')(r_{u_y,i_h} - \overline{r_{u_y}}) \Leftrightarrow B' = (r_{u_a,i_a}' - \overline{r_{u_a}}')(r_{u_y,i_a} - \overline{r_{u_y}}) + \sum_{h=1}^{n'-1} (r_{u_a,i_h} - \overline{r_{u_a}}')(r_{u_y,i_h} - \overline{r_{u_y}})$$

$$B' = dr_{u_a,i_a}(r_{u_y,i_a} - \overline{r_{u_y}}) + (r_{u_a,i_a} - \overline{r_{u_a}}')(r_{u_y,i_a} - \overline{r_{u_y}}) + \sum_{h=1}^{n'-1} (r_{u_a,i_h} - \overline{r_{u_a}}')(r_{u_y,i_h} - \overline{r_{u_y}}) \Leftrightarrow$$

$$B' = dr_{u_a,i_a}(r_{u_y,i_a} - \overline{r_{u_y}}) + \sum_{h=1}^{n'} (r_{u_a,i_h} - \overline{r_{u_a}}')(r_{u_y,i_h} - \overline{r_{u_y}}) \Leftrightarrow B' = dr_{u_a,i_a}(r_{u_y,i_a} - \overline{r_{u_y}}) + B - \sum_{h=1}^{n'} d\overline{r_{u_a}}(r_{u_y,i_h} - \overline{r_{u_y}})$$

$$\Rightarrow e = dr_{u_a,i_a}(r_{u_y,i_a} - \overline{r_{u_y}}) - \sum_{h=1}^{n'} d\overline{r_{u_a}}(r_{u_y,i_h} - \overline{r_{u_y}})$$

Proof of Equation 9

$$C' = \sum_{h=1}^{n'} (r_{u_a,i_h} - \overline{r_{u_a}}')^2 \Leftrightarrow C' = (r_{u_a,i_a}' - \overline{r_{u_a}}')^2 + \sum_{h=1}^{n'-1} (r_{u_a,i_h} - \overline{r_{u_a}}')^2 \Leftrightarrow C' = dr_{u_a,i_a}^2 + 2dr_{u_a,i_a}(r_{u_a,i_a} - \overline{r_{u_a}}') + (r_{u_a,i_a} - \overline{r_{u_a}}')^2 + \sum_{h=1}^{n'-1} (r_{u_a,i_h} - \overline{r_{u_a}}')^2 \Leftrightarrow$$

$$C' = dr_{u_a,i_a}^2 + 2dr_{u_a,i_a}(r_{u_a,i_a} - \overline{r_{u_a}}') + \sum_{h=1}^{n'} (r_{u_a,i_h} - \overline{r_{u_a}}')^2 \Leftrightarrow C' = dr_{u_a,i_a}^2 + 2dr_{u_a,i_a}(r_{u_a,i_a} - \overline{r_{u_a}}') + C + \sum_{h=1}^{n'} d\overline{r_{u_a}}^2 - 2\sum_{h=1}^{n'} d\overline{r_{u_a}}(r_{u_a,i_h} - \overline{r_{u_a}}) \Leftrightarrow$$

$$\Rightarrow f = dr_{u_a,i_a}^2 + 2dr_{u_a,i_a}(r_{u_a,i_a} - \overline{r_{u_a}}') + \sum_{h=1}^{n'} d\overline{r_{u_a}}^2 - 2\sum_{h=1}^{n'} d\overline{r_{u_a}}(r_{u_a,i_h} - \overline{r_{u_a}})$$

Proof of Equation 10

$$D' = \sum_{h=1}^{n'} (r_{u_y,i_h} - \overline{r_{u_y}}')^2 \Leftrightarrow D' = D \Rightarrow g = 0$$

Proof of Equation 11, 12, 13

In the case that user u_y has not rated the item i_a, the values of *B*, *C* and *D* are proved in way similar to equations 8, 9 and 10 respectively. In this case the increments e, f and g equal to:

$$e = -\sum_{h=1}^{n'} d\overline{r_{u_a}}(r_{u_y,i_h} - \overline{r_{u_y}}), \quad f = \sum_{h=1}^{n'} d\overline{r_{u_a}}'^2 - 2\sum_{h=1}^{n'} d\overline{r_{u_a}}'(r_{u_a,i_h} - \overline{r_{u_a}}), \quad g = 0$$