Using Recursive Sorting to Improve Accuracy of Memory-based Collaborative Filtering Recommendations

Serhiy Morozov and Hossein Saiedian
Electrical Engineering and Computer Science
University of Kansas, Lawrence KS, USA

***Abstract*—Modern user behavior datasets contain millions of records, so quickly combining all potentially relevant ratings is often not feasible. Instead, we make suggestions from a small set of the most relevant ratings, so that the memory-based recommender systems could produce simple and accurate results. We propose a new instance selection algorithm that removes irrelevant data after sorting it twice, unlike the traditional approach where the data is only sorted once. The accuracy of the resulting recommendations on the Netflix dataset is considerably better than the standard approach.**

I. Introduction

Collaborative filtering systems recommend items that other users enjoyed, essentially automating "word-of-mouth" suggestions. The assumption is that one user's favorite items may be inferred by observing other users with similar interests. As the name implies, personalized recommendations are derived from filtering all available items through preferences of similar users [13]. However, user opinions are subjective and have little to do with content similarity. In fact, a pure collaborative filtering system has no knowledge of item content, which makes it ideal for abstract domains such as paintings, music, and poetry [13].

Memory-based collaborative filtering is simple and intuitive, does not require many tuning parameters or long training sessions, and can justify recommendations [3], [16]. The recommendation is a consensus of similar users or items, called neighbors. Because some neighbors are more influential than others, this method is often called the K Nearest Neighbors (KNN) approach [18]. To quantify a neighbor's influence, we measure its similarity to the user or item in question, i.e., the active vector [10], [16]. Depending on whether the dataset is a collection of user or item vectors it would be a user- or item-based approach.

The estimate for an active user u on active item a is a weighted sum of neighbors' ratings adjusted by their mean, $\bar{r}_i$.

$$P_{u,a} = \bar{r}_u + k\frac{\sum_{i=1}^{n} w(u,i)(r_{i,a} - \bar{r}_i)}{\sum_{i=1}^{n} w(u,i)}$$

In this formula, n is the neighborhood size, $w(u,i)$ is the influence of a neighbor i, and k is a tuning coefficient [5]. The bottom of this fraction is the sum of the neighbors' weights, which we call net weight. The neighborhood size may vary greatly. Also, some neighbors may offer little influence due to their low similarity. Therefore, the instance selection method that reduces the neighborhood has a large impact on the recommendation accuracy.

When predicting a known rating, the error of an estimate is $P_{u,a} - r_{u,a}$. The accuracy of the entire system may be summarized by the Root Mean Squared Error (RMSE), a popular metric that is especially sensitive to large errors. The goal of this work is to reduce the RMSE.

$$RMSE = \sqrt{\frac{\sum_{u\in U, i\in I} (P_{u,a} - r_{u,a})^2}{|P|}}$$

We evaluate our instance selection algorithms on the Netflix dataset. It contains over 100,000,000 ratings, representing over 17,000 items and over 480,000 users. To establish a point of reference for our experiments, we examine some of the well-known results from the Netflix website, `www.netflixprize.com`. It lists the typical prediction errors of many trivial recommendation approaches that suggest the same rating for every item. For instance, recommending a four star rating for each movie is the most accurate (RMSE = 1.1748), because each recommendation is close to the overall average rating of 3.6 stars. Likewise, recommending 3.6 stars for everyone gives an even smaller error of 1.1287. This value may be reduced further by recommending the movie or user average for each movie and user vector. This results in a typical error of 1.0533 for an average movie and 1.0651 for an average user approach. In general, recommendations with RMSE ¿ 1 are not worth the effort.

II. Recommendation Accuracy Problem

There is a demand for recommender systems that can consistently produce accurate recommendations, but there are few systems that successfully do so. On the one hand, humans are notoriously unpredictable, but on the other hand, there are processing and storage limitations that prevent extensive dataset analysis. Furthermore, user behavior data is not perfect, and there is usually little of it. However, it is often the only source of information available, so we need a way to infer user behavior patterns from sparse data. Many studies show dramatic recommendation quality improvements due to changes in the data [5], [9], [17]. In a sense, optimizing anything else is comparable to fixing the symptoms, while improving the data addresses the root of a problem.

The easiest way to reduce data sparsity is to add default ratings to the dataset. Some of the simplest default ratings include mean rating and majority rating [5]. However, aggregate defaults are usually poor approximations for the actual opinions [9], so they are often made neutral or negatively skewed to ensure a more conservative prediction.

Sometimes, default ratings come from external sources of actual opinions. For example, the MovieLens project populated missing values with existing ratings from a different movie dataset [17]. Likewise, Basu, Hirsh, and Cohen used the Internet Movie Database website to supplement their dataset [2]. Using external sources of default ratings is a simple and effective way to reduce data sparsity, but those sources may not always be available.

Missing data can also be inferred from a cluster of similarly classified vectors. For example, the GroupLens project clustered users based on their preferences and related news articles based on their topic [12]. As a result, each cluster appeared to have a rich rating history. Such clusters provide good default ratings, but require a way to determine vector membership. In order to group users with no expressed preferences, some systems consider additional properties like age, gender, and education [1]. Item clusters often use domain-specific knowledge, which is rarely available and may not have clear-cut boundaries. Clustering is an effective way to reduce sparsity and improve performance, but it sacrifices personalization.

One way to improve the quality of data, without supplementing the dataset, is to remove unnecessary ratings. Data reduction algorithms shrink a dataset without damaging it. They preserve the useful information, remove noise, and decrease the amount of computation necessary to complete a recommendation [7], [11]. Such algorithms are especially relevant when scaling up is not an option.

Instance selection is a common data reduction technique that has been traditionally used for data classification. The instance selection algorithm chooses the smallest possible portion of the available data, such that a successful classification may still occur [7], [11]. For example, some algorithms remove instances that do not affect other classifications [14] and some employ a ranking mechanism to eliminate irrelevant instances [6]. A memory-based recommendation is essentially a way to classify one's opinion, given a set of friends' opinions and a way to quantify their influence. Therefore, instance-based learning algorithms, like KNN, could benefit from instance selection.

Even though instance selection algorithms are meant to manage an overwhelming amount of data, the algorithms themselves do not scale well. In fact, most approaches exhibit at least quadratic complexity, which makes them unsuitable for many serious applications [7], [8]. Therefore, most instance selection algorithms do not apply to the problems that would benefit most from their use. However, recommendation accuracy improvement is considerable and we focus our research on this aspect of instance selection.

III. Instance Selection Algorithms

Before making a recommendation, we first identify all potentially relevant ratings. We locate all users who rated the active item and all items that were rated by the active user. Based on these two lists, we locate a set of ratings that are either on a related item (according to the active user) or given by a related user (according to the active item). The ratings are then placed in a matrix where items are rows and users are columns. Each cell contains a rating that is associated with a single user and an item. We refer to rows in such a matrix as item vectors and columns as user vectors. A transposed matrix would represent users as rows and items as columns, so we can produce user- and item-based recommendations from the same data.

The standard instance selection approach ranks neighbors based on every dimension. It compares every row of the matrix to the active vector and rearranges the rows in the order of decreasing similarity. As a result, the most influential vectors are concentrated at the top of the matrix because the active vector is the first row. Truncating such a matrix deletes only the least relevant data.

The recursive instance selection approach also sorts and truncates the matrix, except it does so in two passes. The first pass sorts and identifies the top 30 most similar dimensions. The second pass selects vectors with the highest similarities according to these dimensions. A single sort can establish the most similar dimensions, but not the nearest neighbors according to those dimensions. The recursive approach chooses the best neighbors according to the best dimensions.

Both methods eventually truncate the matrix to 30 rows, i.e., a neighborhood of at most 30 most similar vectors. Empirical results show this size to be particularly accurate [15], [16]. A truncated matrix contains enough data to establish the mean of a vector, which is used to support a particular recommendation, yet does not introduce unnecessary noise that causes over-fitting.

IV. Recursive Sorting Rationale

Collaborative filtering assumes that users who agreed in the past are likely to agree in the future. As a result, users who agree more tend to have a bigger influence. Ideally, the opinions are unanimous and every neighbor has a weight of 1. In the worst case scenario, everyone's weight is 0, which means that neighbors have no shared dimensions. Therefore, the total influence in a neighborhood of n vectors is between 0 and n. In reality, the net neighborhood weight is somewhere in the middle, because both instance selection algorithms require at least one shared dimension for all neighbors. Maximizing the net weight of a fixed size neighborhood more closely resembles the best case scenario.

Since neighborhoods are restricted to a fixed size, the only way to guarantee high net weights is to consider the most influential neighbors first. The first n vectors ordered by their influence will always produce a net weight greater than that of a random sample of neighbors. Sorting ensures that after truncation the neighborhood contains most similar vectors as opposed to a random sample of them. Truncating the neighborhood without sorting it first may still produce a high net weight, but such outcome is unlikely.

Another fundamental assumption of collaborative filtering is that similar users agree on most items, regardless of their domain. We refer to this type of comparison as "global similarity" because all common dimensions contribute to the similarity of any two vectors. Global similarity can identify very influential neighbors, which are extremely rare. Our approach requires two users to agree on a few of the most relevant items, not all of them. We refer to this type of comparison as "local similarity" because only a subset of all common dimensions contributes to the similarity.

Standard approach uses global similarity to rank vectors. Recursive approach first identifies a subset of dimensions and then ranks the vectors by their local similarity on those dimensions. For instance, two globally similar users may disagree on a few movies. As long as the number of such movies is sufficiently small, the global similarity remains high. However, local similarity according to these movies would conclude the two users to be less similar. Likewise, one may compare the two globally dissimilar users across the commonly liked movies and get a high local similarity.

Consider the following examples that demonstrate the changes in global and local similarity between two users. We use cosine similarity to quantify the strength of a relationship between two vectors. It is the baseline metric for many collaborative filtering systems [15]. In this case, both users have rated the same three movies. However, the similarity between user vectors may be established across all or just the first two common dimensions. In fact, the choice of common dimensions has a large effect on the perceived vector similarity.

$$u_1 =< 1, 2, 3 >; u_2 =< 1, 3, 1 >; cos(u_1, u_2) = 0.806$$
$$u'_1 =< 1, 2 >; u'_2 =< 1, 3 >; cos(u'_1, u'_2) = 0.990$$

$$u_1 =< 1, 2, 3 >; u_2 =< 5, 1, 3 >; cos(u_1, u_2) = 0.723$$
$$u'_1 =< 1, 2 >; u'_2 =< 5, 1 >; cos(u'_1, u'_2) = 0.614$$

Comparing two vectors on fewer dimensions could produce a higher similarity if the dimensions are sorted. Consider a case where we sort the matrix by rows and columns such that the most similar vectors are positioned closer to the top left corner of the matrix. Assuming that dimensions are sorted, comparing vectors on less similar dimensions will allow the outliers on a neighbor's rating scale to affect the similarity metric. Extreme opinions on different scales are less likely to agree, so the similarity of such vectors would decrease. If this is false, then considering an extra dimension will result in higher vector similarity, i.e., most of the rows agree on this column. If that were the case, the additional columns should be considered earlier, since columns for which the rows agree most often are placed first. However, this is impossible because the columns are considered in order of decreasing similarity.

To demonstrate this principle, consider five vectors with five dimensions $< a, b, c, d, e >$ in Figure 1. The net weight of such matrix is 3.56, where the weight of each vector is quantified by its Pearson's correlation to the active vector. In other words, we sum the similarities of the 1st row and the 2nd row, 1st row and 3rd row, 1st row and 4th row, etc. Then we delete one of the columns and recompute the similarities again. The net weights of four truncated versions of this matrix, with one of the dimensions removed, are as follows: no e = 3.87, no d = 4.08, no c = 3.38, no b = 2.94. Removing some dimensions increases the net weight, but how does one know which dimensions to remove?

$$\begin{array}{ccccc}
a & b & c & d & e \\
3 & 4 & 2 & 1 & 3 \\
2 & 3 & 1 & 1 & 3 \\
3 & 4 & 3 & 3 & 4 \\
4 & 4 & 2 & 2 & 4 \\
3 & 3 & 2 & 3 & 2
\end{array}$$

Fig. 1. A 5×5 Matrix

Consider the similarity of each dimension to a: $a = 1.00, b = 0.65, c = 0.50, d = 0.35, e = 0.42$. The d and e dimensions are the least similar and removing them increases the net weight. Figure 2 shows the negative correlation between the similarity of a dimension and the net weight of a truncated matrix that does not contain it.

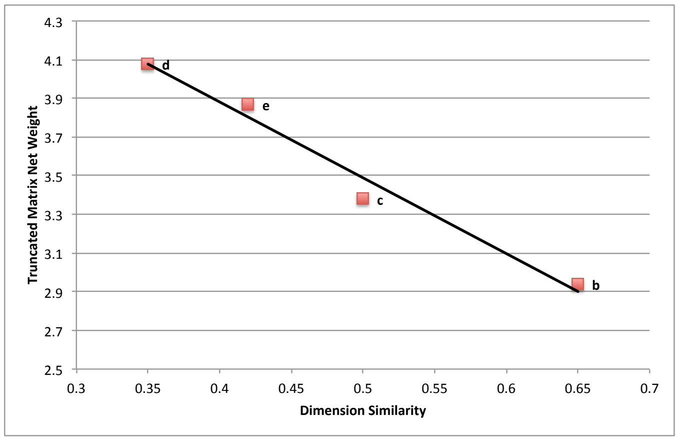

Fig. 2. Truncated Matrix Net Weights

To verify this phenomenon, we examined the neighbors identified by the two algorithms on the Netflix dataset. Figures 3 and 4 show typical similarities in an item and user-oriented neighborhood. In both cases, the recursive approach identifies neighbors with higher similarities and greater net weight, i.e., area under the curve.

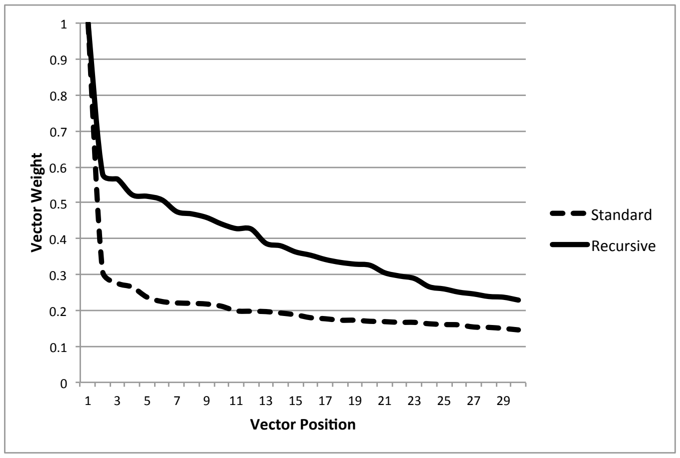

Fig. 3. Item Similarity Comparison

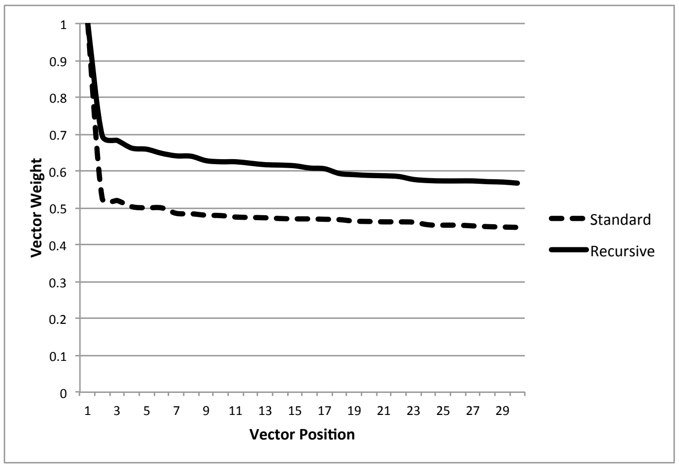

Fig. 4. User Similarity Comparison

V. Justification for Resorted Data

To further support the benefits of our recursive algorithm, we considered a statistical justification of this approach. The Rao-Blackwell theorem states that if $g(x)$ is an estimator for θ, then conditional expectation of $g(x)$ given a sufficient statistic $T(x)$ is a better estimator of θ and never worse [4]. This theorem employes a well-known relationship between conditional and unconditional variance, i.e., $var(E(g(x)|T(x))) <= var(E(g(x)))$. Smaller variance means smaller Mean Squared Error, which means higher overall system accuracy. Therefore, we can improve recommendation accuracy by employing estimators which are functions of the sufficient statistic. In other words, we need an instance selection algorithm that represents a sufficient statistic of the data.

In the context of our recommender system, x is a set of ratings in the dataset, $g(x)$ is the influence of a neighbor, $E(g(x))$ is the weighted average of neighbor's opinions scaled by their influence, $T(x)$ is a sufficient statistic computed from the original data, and $E(g(x)|T(x))$ is the conditional expected value of a rough estimator given a sufficient statistic. In other words, it is a weighted average of all ratings that have the same value for the sufficient statistic, i.e., local weights on dimensions selected by their global similarity ranking.

A sufficient statistic is a function of the data that describes it in such a way that a sample generated according to this statistic would be as useful as the original data for estimating θ, the actual rating we are trying to predict. The purpose of the sufficient statistic is to capture all of the useful information necessary for estimating θ, so that the data may be discarded in favor of the statistic. The list of the most relevant dimensions, established in the first step of the recursive algorithm, is a sufficient statistic of the data. Once we know the best dimensions, we no longer need the rest of them.

The first pass of the recursive algorithm decides which dimensions are the most relevant. It categorizes the matrix dimensions into two groups: top 30 and everybody else. Vectors from the first group receive a weight of $T(x) = 1$ and everyone else receives a weight of $T(x) = 0$. Ignoring dimensions from the latter group may improve the accuracy of an existing estimator $g(x)$. In fact, Rao-Blackwellisation of $g(x)$ is guaranteed not to make things worse.

We considered two alternative instance selection methods with cosine similarity as well as Pearson's correlation measures on 1,000 randomly chosen ratings from the Netflix Quiz dataset. Figure 5 shows that using Pearson's correlation is considerably better than cosine similarity. In fact, this approach had lower RMSE scores for both vector orientations. Also, resorting the data was more accurate for cosine as well as Pearson's similarities.

Even though Pearson's correlation is a more accurate way to compare vectors, the choice of a similarity measure is irrelevant for our instance selection process. The benefits of our approach come not from a particular weight metric, but from reducing the number of dimensions and deciding which dimensions should participate in the similarity computation.

Method	KNN-Item	KNN-User
Cosine Standard	1.301	1.305
Cosine Recursive	0.980	0.945
Pearson Standard	0.786	0.826
Pearson Recursive	0.423	0.465

Fig. 5. RMSE on the Netflix Quiz Dataset

VI. Conclusion

We believe that a small number of relevant ratings is sufficient to make an accurate recommendation. Such ratings may be chosen with a recursive approach that requires two neighbors to be similar in some, but not all, domains. It establishes more pertinent evidence for vector similarity, so that selected ratings are more relevant. To test this claim, we developed an algorithm that organizes ratings in matrices sorted by user/item similarities.

The main purpose of our instance selection algorithm is to produce small and dense matrices. It selects relevant data by recursively resorting the matrix, since local similarities of users and items are mutually dependent. The resulting vectors do not necessarily agree in every shared dimension, but they hold the most insight about the current recommendation. Our analysis shows that resorting the matrix can not decrease recommendation accuracy. Furthermore, our empirical study shows that the recommendation accuracy from resorted matrices is considerably better than the standard approach where the data is only sorted once.

References

[1] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734–749, 2005. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1423975

[2] C. Basu, H. Hirsh, and W. Cohen, "Recommendation as classification: Using social and content-based information in recommendation," in *Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*. Menlo Park, CA, USA: American Association for Artificial Intelligence, 1998, pp. 714–720.

[3] R. M. Bell and Y. Koren, "Lessons from the netflix prize challenge," *ACM SIGKDD Explorations Newsletter*, vol. 9, no. 2, pp. 75–79, 2007.

[4] D. Blackwell, "Conditional expectation and unbiased sequential estimation," *The Annals of Mathematical Statistics*, vol. 18, no. 1, pp. 105–110, 1947. [Online]. Available: http://www.jstor.org/stable/2236107

[5] L. Candillier, F. Meyer, and M. Boullé, "Comparing state-of-the-art collaborative filtering systems," in *Proceedings of the 5th International Conference on Machine Learning and Data Mining in Pattern Recognition*, ser. LNCS, vol. 4571. Springer, 2007, pp. 548–562.

[6] C. de Santana Pereira and G. Cavalcanti, "Instance selection algorithm based on a ranking procedure," in *The 2011 International Joint Conference on Neural Networks (IJCNN)*, 31 2011-aug. 5 2011, pp. 2409 –2416.

[7] C. García-Osorioa, A. de Haro-Garcíab, and N. García-Pedrajasb, "Democratic instance selection: A linear complexity instance selection algorithm based on classifier ensemble concepts," *Artificial Intelligence*, vol. 174, no. 5-6, pp. 410 – 441, 2010.

[8] N. García-Pedrajas, J. A. Romero Del Castillo, and D. Ortiz-Boyer, "A cooperative coevolutionary algorithm for instance selection for instance-based learning," *Mach. Learn.*, vol. 78, pp. 381–420, March 2010. [Online]. Available: http://dx.doi.org/10.1007/s10994-009-5161-3

[9] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Transactions on Information Systems*, vol. 22, no. 1, pp. 5–53, 2004.

[10] Z. Huang, H. Chen, and D. Zeng, "Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering," *ACM Transactions on Information Systems*, vol. 22, no. 1, pp. 116–142, 2004.

[11] N. Jankowski and M. Grochowski., "Comparison of instances selection algorithms: I. algorithms survey," in *Artificial Intelligence and Soft Computing*, ser. Lecture notes in computer science. Springer, June 2004, pp. 598–603.

[12] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl, "GroupLens: Applying collaborative filtering to usenet news," *Communications of the ACM*, vol. 40, no. 3, pp. 77–87, 1997.

[13] N. Leavitt, "Recommendation technology: Will it boost e-commerce?" *Computer*, vol. 39, no. 5, pp. 13–16, 2006.

[14] E. Marchiori, "Class conditional nearest neighbor for large margin instance selection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, pp. 364–370, 2010.

[15] N. Miller, Bradley, A. Konstan, Joseph, and J. Riedl, "PocketLens: Toward a personal recommender system," *ACM Trans. Inf. Syst.*, vol. 22, no. 3, pp. 437–476, 2004.

[16] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th International Conference on World Wide Web*. New York, NY, USA: ACM, 2001, pp. 285–295.

[17] B. M. Sarwar, J. A. Konstan, A. Borchers, J. Herlocker, B. Miller, and J. Riedl, "Using filtering agents to improve prediction quality in the GroupLens research collaborative filtering system," in *Proceedings of the 1998 ACM Conference on Computer Supported Cooperative Work*. New York, NY, USA: ACM, 1998, pp. 345–354.

[18] J. Wang, A. P. de Vries, and M. J. T. Reinders, "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*. New York, NY, USA: ACM, 2006, pp. 501–508.