Collaborative Filtering using Electrical Resistance Network Models with Negative Edges

Jérôme Kunegis and Stephan Schmidt

DAI-Labor, Technische Universität Berlin, Franklinstraße 28, 10587 Berlin, Germany
{jerome.kunegis, stephan.schmidt}@dai-labor.de

Abstract. In a recommender system where users rate items we predict the rating of items users have not rated. We define a rating graph containing users and items as vertices and ratings as weighted edges. We extend the work of [1] that uses the *resistance distance* on the bipartite rating graph incorporating negative edge weights into the calculation of the resistance distance. This algorithm is then compared to other rating prediction algorithms using data from two rating corpora.

1 Introduction

In recommender systems items are recommended to users. Collaborative filtering is a common approach to implementing recommender systems. One way of implementing a recommender system is by collaborative filtering. Instead of calculating an item score based on item features, a collaborative filtering algorithm analyzes existing ratings from users to predict the rating of an item a certain user has not seen.

Items are typically documents, songs, movies or anything that can be recommended to users and that users can rate. Ratings can be given by users explicitly such as with the five star scale used by some websites or can be collected implicitly by monitoring the users' actions such as recording the number of times a user has listened to a song.

The approach presented here models rating databases as bipartite graphs with users and items as the two vertex sets and ratings as weighted edges. [1] describes how this graph can be seen as a network of electrical resistances and how the total equivalent resistance between any two nodes can be used to define a similarity function either between two users, two items or users and items. In the referenced paper edges are not weighted and thus all resistances are modeled as unit resistances.

We extend this work by using the actual ratings as edge weights. Because ratings can be negative, modifications to the previous approach are necessary in order for the similarity function we define to satisfy three basic conditions, which we present in the section defining the rating graph. We use this similarity for predicting ratings and compare the results to common prediction algorithms.

Based on the graph representation of a rating database we can formulate three conditions a good prediction measure should satisfy:

Parallelity Parallel paths of edges between two nodes contribute monotonically to the similarity between the two nodes. If multiple paths exist between two nodes, the overall similarity between the two nodes should be greater than the similarity taken on each path separately.

Transitivity A long path of edges with positive weight results in a lower similarity than a short path with similar weights.

Negative ratings A path consisting of both positive and negative edges leads to a negative similarity exactly if the number of negative edges is odd. This follows from the observation that users that both like or both dislike an item are similar while two users are not similar when one of them like an item the other one dislikes.

We will show later that our method satisfies all three conditions while the original algorithm [1] only satisfies the first two.

Outline. In Section 2 we present mathematical definitions and the two basic rating prediction methods: rating normalization and weighted rating sum based on the Pearson correlation between users. Section 3 defines the bipartite rating graph and the meaning of the rating values. Section 4 defines the similarity between two graph nodes based on electrical resistances in the case that no ratings are negative. In Section 5, the usage of negative ratings is explored and a modified formula is presented that satisfies the three conditions presented above. Section 6 discusses algorithms for solving the system of equations. In Section 7 our similarity measure is compared to two standard prediction measures and to the prediction measure defined in [1]. Section 8 finishes with some remarks about the accuracy of our method and with future work.

2 Related Work

This section presents the two prediction methods used in our evaluation that are not based on the rating graph. The method of [1] is presented after the definition of the rating graph.

2.1 Definitions

Let $\mathcal{U} = \{U_1, U_2, \ldots, U_m\}$ be the set of users and $\mathcal{I} = \{I_1, I_2, \ldots, I_n\}$ the set of items.

The rating by user U_i of item I_j is denoted r_{ij}. We keep r_{ij} undefined if user U_i has not rated item I_j.

Let $\mathcal{I}_i$ be the set of items rated by user U_i and $\mathcal{U}_j$ the set of users that rated item I_j.

2.2 Rating Scales

Different rating systems use different rating scales. When user ratings correspond to labels such as *good* or *bad*, they are represented by an integer value ranging for instance from 1 to 5, where 5 represents the highest rating (*very good*), and 1 represents the lowest rating (*very bad*). Users can give neutral ratings when the number of different values is odd. These ratings can be adjusted such that a neutral value is represented by the value 0. They can also be scaled to fit in the $[-1, +1]$ range, but this is not necessary because the data is normalized as described below as a first step in most algorithms. In some cases ratings may be unbounded, for instance if they represent a document view count.

2.3 Normalization

Different users have a different idea of what *good* and *bad* ratings mean, so some users give the extremal ratings rarely while others always rate items as either very good or very bad. Assuming that the distribution of item ratings should be similar for all users we can scale each user's ratings around the mean rating such that the standard deviation of each user's ratings is a constant. Using the same argument, we can offset each user's ratings so that his mean rating is zero. This kind of transformation is described in [2].

2.4 Mean Rating

As the simplest prediction algorithm, we chose the user's mean rating as a prediction for his rating of any item he has not rated. This is equivalent to always predicting a rating of zero after normalization. For each user U_i, we define the mean rating $\bar{r}_i$:

$$\bar{r}_i = \frac{1}{|\mathcal{I}_i|} \sum_{j \in \mathcal{I}_i} r_{ij}$$

This method is very simple as it does not take into account other users' ratings.

2.5 Pearson Correlation

To take other users' ratings into account, a simple method to predict the rating for a user $U \in \mathcal{U}$ is to calculate the mean rating other users have given to the item in question. However, other users may have a different taste than user U. Therefore we need a similarity function applicable between user U and other users. This similarity value can then be used as weights to get a weighted mean as a rating prediction. The similarity function used can even admit negative values, indicating that users are likely to have opposing tastes.

A similarity function satisfying these conditions is the Pearson correlation between users calculated using normalized ratings as described in [3]. This correlation is normally calculated using only items the two users have both rated. As described in [3], it is also possible to take the correlation over all items rated by at least one user, filling missing ratings with a default value.

Let $\mathcal{I}_{ab} = \mathcal{I}_a \cap \mathcal{I}_b$. The Pearson correlation between the users U_a and U_b is defined as

$$w(a,b) = \frac{\sum_{j \in \mathcal{I}_{ab}} (r_{aj} - \bar{r}_a)(r_{bj} - \bar{r}_b)}{\sqrt{\sum_{j \in \mathcal{I}_{ab}} (r_{aj} - \bar{r}_a)^2 \sum_{j \in \mathcal{I}_{ab}} (r_{bj} - \bar{r}_b)^2}}$$

where $\bar{r}_a$ and $\bar{r}_b$ are taken over $\mathcal{I}_{ab}$. The rating prediction of item U_j for user U_i is now given by

$$r_{ij}^p = \left(\sum_a w(i,a) \right)^{-1} \sum_a w(i,a) r_{aj} \tag{1}$$

where sums are taken over all users that have rated items in common with user U_i. This expression is not defined when the sum of correlations is zero.

Alternatively, we can use all items at least one user has rated, and substituting these missing ratings with a neutral value such as the user's mean rating. Throughout this paper only the first option will be used as it is common practice in collaborative filtering recommender systems.

3 The Rating Graph

In this section, we define the weighted rating graph. Given sets of users, items and ratings, users and items are represented by vertices, while ratings are modeled as the edges connecting them. Rating values become edge weights.

As observed by [4] and [5], this rating graph is bipartite. We can assume that the graph is connected. If it is not, it can be made connected by only keeping the connected component containing the vertices we want to compare. If the vertices to be compared are not connected, then the graph cannot be used to predict the rating in question since the vertices are not part of the same rating subgraph. In fact, the absence of a path between two nodes means that there is no way to say anything about the relation between the two users or items, and no algorithm or model can give results in this case.
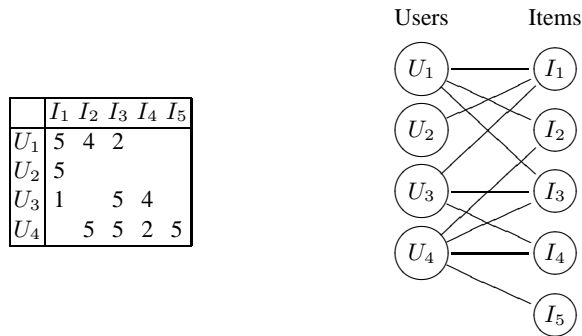

	I_1	I_2	I_3	I_4	I_5
U_1	5	4	2		
U_2	5				
U_3	1		5	4	
U_4		5	5	2	5

Fig. 1. User-item rating table and graph, on a scale from 1 (very bad) to 5 (very good)

In Figure 1 four users (U_1–U_4) have rated five items (I_1–I_5). However, not all possible ratings were given. Empty cells in the rating table on the left denote ratings users have not given.

The corresponding bipartite rating graph is shown on the right. It contains one edge for each rating. In this case, the graph is connected.

As explained in the introduction, the purpose of the graph is to find connections between users and items. As an example for the graph given above, we could ask the question: How would user A rate item I_5? Since the user has not rated the item, we must use the known ratings to predict a rating for the corresponding vertex pair.

Because users and items are both represented as vertices, our method for calculating a similarity them can also be used for calculating similarities of two users or two items. In this paper, we restrict ourselves to calculating the similarity between a user and an item.

In the case covered in [1], the graph is not bipartite, but tripartite: Vertices represent users, movies that can be rated, and categories of movies (such as comedies). In the resulting tripartite graph however, the problems given in [1] can all be modeled as the similarity between two vertices. Therefore, we will consider the general case of graphs that need not necessarily be bipartite.

Given a weighted graph and two vertices, a similarity measure between the two vertices can now be defined. As mentioned before, edges are weighted, and their weights are signed. Positive values indicate a positive association between the nodes, negative values indicate a negative association between the nodes. The measure we want to define must satisfy the three conditions presented in the introduction. These conditions translate to the following mathematical properties:

Long paths Paths from one vertex to another can be compared by the number of edges they contain. Long paths must correspond to smaller similarities than short paths. If, for instance, two users have rated the same item, then their similarity must be higher than if they had only rated items in common with a third user. However, long paths must not give a similarity of zero. This condition may be called transitivity.

Parallel paths If a large number of distinct paths exist between two vertices, the similarity value between the two vertices must be higher than for the case where only few paths exist.

Negative edges If the two vertices are connected by exactly one path, then the resulting similarity must be of the same sign as the product of the weights of the path edges. This condition corresponds to the sensible assumption that one will dislike items that are disliked by users with similar taste.

We will now show that the Pearson correlation based rating prediction does not satisfy all three conditions.

In the bipartite rating graph, the Pearson correlation can only be calculated between two user vertices that are connected by a path of length two. If the distance between the two vertices is longer than two edges, the Pearson correlation cannot be calculated because the users have not rated any items in common. Thus, the Pearson correlation does not satisfy the requirement on long paths. However, it does satisfy the requirement on parallel paths and the requirement on negative edges.

In Figure 2, the Pearson correlation between users U_1 and U_2 is calculated. Edges that are not taken into account during calculation are shown as dotted lines. This example shows that while both user U_1 and user U_2 have something in common with user U_3, these connections are simply ignored by the Pearson correlation. Furthermore, user U_4 has no ratings in common with user U_1, so the Pearson correlation cannot be calculated between these two users.
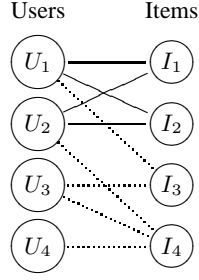
Users   Items

Fig. 2. In this example, the edges not taken into account by the Pearson correlation calculation are shown as dotted lines.

4   Resistance Distance

This section describes the similarity function based on electrical resistance as described in [1].

We have previously proposed that a sensible similarity measure needs to be smaller as paths get longer and larger when there are parallel paths. A similar behavior can be encountered in electrical engineering regarding electrical resistances: When in series, their values add to each other; when in parallel, the inverse of their values add to each other, and the resulting resistance is lower than individual resistances.

Our function is required to yield the opposite result: A path of edges in the rating graph (corresponds to resistances in series) must result in a lower value. Similarly, parallel paths must lead to a rating value that is the sum of the individual path values. Therefore, we use the inverse of the resistance in our function. The inverse of the electrical resistance is the electrical conductance. In this paper, we use the letter r to denote ratings, which must not be confused with the letter R that usually denotes resistances.

In [1] all ratings are initialized with the unit rating ($r_{\mathrm{unit}} = 1$) to avoid negative ratings, which are problematic as we will see below. Figure 3 shows some examples containing only unit resistances (or, equivalently, unit conductances). The corresponding resulting conductance is given for each graph, illustrating how the length of paths influences the similarity value, and how parallel edges result in higher similarity values. As mentioned in [6], the resistance distance between nodes of a graph is a metric.

We will now describe a method for calculating this total conductance between vertices A and B. When applying a unit voltage on the two vertices, the current through the network will equal the total conductance. In order to compute the value of the current passing through the network, we first need to calculate the potential on A and on the vertices adjacent to A. To this end, we introduce a variable x_V for each vertex V. We then simulate the application of a unit voltage between A and B by setting $x_A = 0$ and $x_B = 1$. For each vertex V adjacent to the vertices $V_1 \ldots V_k$, we know that the total current entering V must be zero. Let I_i be the current going from V_i to V and r_i the conductance of the edge (V, V_i). We obtain the following system of equations:

$$\sum_i I_i = 0$$

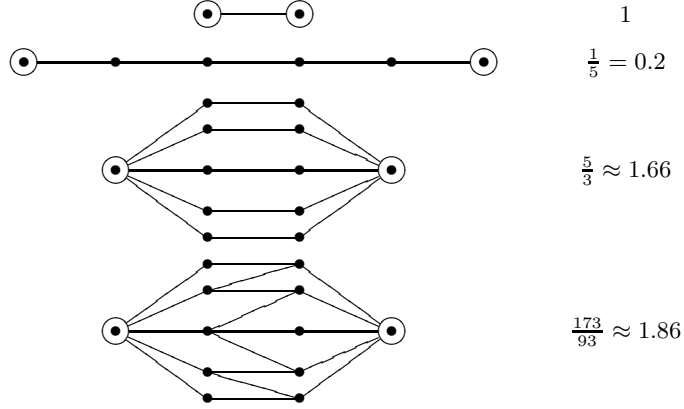**Fig. 3.** Bipartite rating graphs annotated with resistance distance values between pairs of highlighted nodes. All edges have unit weight.

$$\sum_i r_i(x_V - x_{V_i}) = 0$$

$$\sum_i r_i x_V - r_i x_{V_i} = 0$$

$$\left(\sum_i r_i\right) x_V = \sum_i r_i x_{V_i} \tag{2}$$

At this point, an additional variable is introduced for each vertex in the graph. Solving this system of equations will yield potentials for all vertices of the graph. The current flow from A to B is now given by taking the sum of the current over all edges (A, V_i) incident to A:

$$I_{AB} = \sum_i I_{AV_i}$$

$$= \sum_i r_{AV_i}(x_{V_i} - x_A)$$

$$= \sum_i r_{AV_i} x_{V_i}$$

The resistance distance is now given by:

$$r_{\text{eq}} = \frac{I_{AB}}{x_B - x_A}$$

$$= I_{AB}$$

$$= \sum_i r_{AV_i} x_{V_i} \tag{3}$$

And inverse resistance distance gives our similarity between two nodes A and B:

$$\text{sim}_{\text{unit}} = r_{\text{eq}}^{-1}$$

We observe that finding the total conductance between A and B is equivalent to solving a linear system of n equations and n variables, where n is the number of vertices in the rating graph.

5   Negative Ratings

As we have seen, the inverse resistance distance satisfies the first two conditions we imposed on similarity functions. However, it is easy to see that it does not conform to the third condition; for instance in a path containing an odd number of negative edges, the similarity should be negative, but the inverse resistance distance is always positive.

Instead of just setting all ratings to the value one to avoid negative edge weights, we will try to keep the real values, and try to define a new similarity measure satisfying all three conditions.

We now examine the consequences of inserting the *original*, possibly negative rating values into the equations above. Figure 4 shows a very simple electrical network with negative resistances, and its corresponding inverse resistance distance is calculated using the formula known from physics stating that two resistance in series of magnitude r_1 and r_2 are equivalent to a single resistance of magnitude $\frac{r_1 r_2}{r_1 + r_2}$.

$$\bullet \xrightarrow{\;r_1 = +1\;} \bullet \xrightarrow{\;r_2 = -1\;} \bullet \qquad r_{\text{eq}} = \frac{r_1 r_2}{r_1 + r_2} = \frac{1 \cdot (-1)}{1 + (-1)} = \frac{-1}{0}$$

Fig. 4. The inverse resistance distance on graph with negative edge weights may not be defined.

As can be seen, the formula leads to a division by zero. However, we require the result value to be smaller than 1 in absolute value (from the first condition) and negative (from the third condition). The desired result is obtained in a different way; the absolute value of our result is equal to the value calculated using the formula containing the absolute values of the conductances. The sign of the result then has the sign of the product of the two ratings. Thus we want:

$$r_{\text{eq}} = \text{sgn}(r_1)\text{sgn}(r_2)\frac{|r_1| \cdot |r_2|}{|r_1| + |r_2|} = \frac{r_1 \cdot r_2}{|r_1| + |r_2|}$$

Figure 5 displays some examples and their resulting similarities using this formula.

Now, the question arises how the general equations given in Equation 2 are to be adapted to yield the desired result for our simple example cases. If we interpret an edge with rating $-r$ not as a negative resistance of conductance $-r$, but as a resistance with conductance r that "inverts" the potential difference across the resistance, then on the right side of our equation the factor of x_{V_i} remains r_i, but in the sum on the left side the absolute value of r_i must be used. This means that for each vertex V, we replace the equation
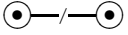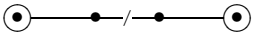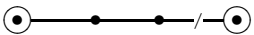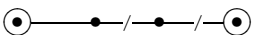
$$\left(\sum_i r_i\right) x_V = \sum_i r_i x_{V_i}$$

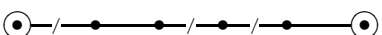| Rating Graph	Similarity

Fig. 5. Rating graphs with negative edges and the similarity between the highlighted nodes calculated taking into account the negative values. Edges with a slash have weight -1, other edges have weight $+1$.

with

$$\left(\sum_i |r_i|\right) x_V = \sum_i r_i x_{V_i}$$

For the examples of Figure 5, these equations yield the desired results. In the next section, methods for solving the resulting system of equations are given.

6 Y-Δ Simplification

Calculating the resistance distance between two nodes in a network involves solving a system of n equation of n variables, where n is the number of nodes in the network. Before solving a system of equations however, we can try to simplify the network without changing the resistance distance. The examples in Figure 6 show simple cases where this is possible.

In case a), node D has degree 2. No current can enter this node, so nodes D and B always have the same potential. Therefore, the conductance r_{ab} of the edge (B, D) has no influence on the resulting conductance. We can just remove this edge without changing the resulting conductance.

In case b), node B has degree 2. This node and its two incident edges can be replaced with a single edge whose resistance is the sum of the two original edges. In this case, the resulting edge weight may become greater than 1 in absolute value.

In the cases a) and b), we have simplified the graph by removing one vertex, removing its incident edges, and in case b) adding one additional edge. In case c) however, the two vertices that could be removed have degree three, so we cannot apply one of the two simplifications as before. Instead, we can use the so-called Y-Δ simplification. We replace node B and its three incident edges (that form a Y) with a triangle of three edges (that form a Δ). Figure 7 shows the Y-Δ transformation graphically. The formulas giving the new resistance values are fundamental to the theory of electrical resistances, as given e.g. in [7].
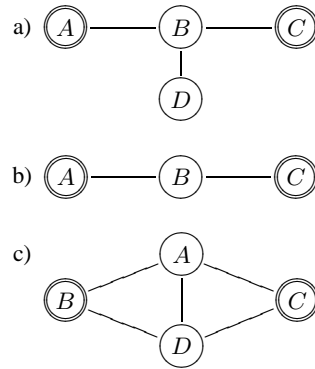
Fig. 6. Simple cases of vertex elimination. The similarity is to be calculated between the doubly circled nodes.
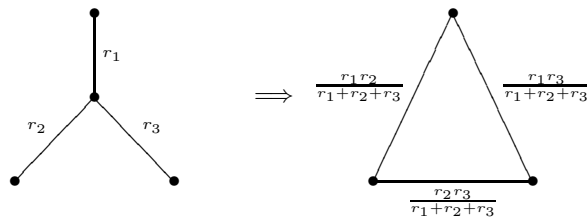

Fig. 7. Principle of Y-Δ simplification

Note that if an edge of the triangle already exists in the graph prior to Y-Δ transformation, the resulting graph will have parallel edges. As parallel conductances are additive, we can just add the new edge value to an existing edge value if necessary.

As further shown in [7], Y-Δ simplification can be generalized to removal of vertices of any degree. In the general case, vertex A of degree k adjacent to vertices $\{B_1, \ldots, B_k\}$ with resistance r_i between A and B_i are replaced by $\binom{k}{2}$ new edges. Between each pair of vertices (B_i, B_j), a new edge is added with weight $\frac{r_i r_j}{\sum_k r_k}$. Figure 8 shows an example for $k = 4$.
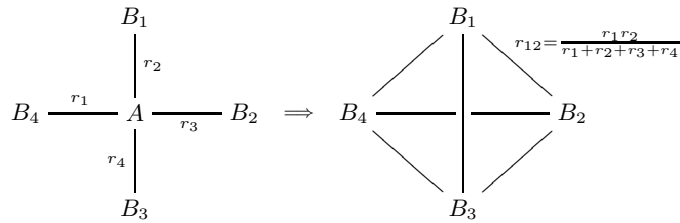

Fig. 8. Elimination of a vertex with degree four

While this generalized removal of nodes works in the simple case of only positive conductance values, it does not work with our modified total conductance. Figure 9 gives an example where according to our definition, the conductance between A and B is $1/5$, but by using simplification of nodes, we get the result $1/3$.
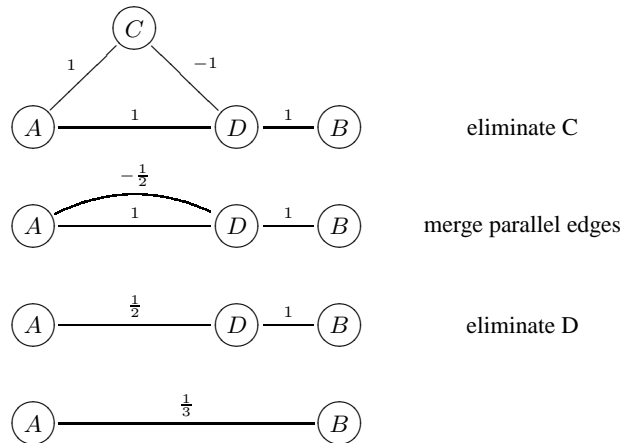

Fig. 9. Simple vertex elimination results in different similarity value than defined.

This result, however, does not correspond to the system of equations presented above. The following system of equations must be solved to get the value of the total conductance between A and B according to our definition:

$$x_A = 0$$
$$x_B = 1$$
$$2x_C = x_A - x_D$$
$$3x_D = x_A + x_B - x_C$$

Solving this system, we get $r_{\mathrm{eq}} = x_C + x_D = 1/5$. This example shows that Y-Δ elimination can not be applied in the case of modified resistance distance calculation. Therefore, the system of equations defined by Equation 2 must be solved. As shown in [8] and [9], this system of equations is sparse when the rating matrix is sparse, and this is the case in practice because each user will only rate a small number of items compared to the total number of items available.

As mentioned in [9], the minimum degree algorithm has a runtime of $O(n^3)$, where n is the number of users and items. In the typical case, many users have rated only very few items, and many items were only rated by few users. Because the corresponding nodes have small degrees, they are eliminated first, and in this phase of the algorithm, the runtime is linear.

7 Evaluation

We use two corpora for evaluation: MovieLens [1] and Jester [2]. Each evaluation test consists of the following steps: First we choose a rating at random, then we remove this rating from the corpus. Afterward, we use a prediction algorithm to predict that rating. These steps are repeated for each of the two corpora, and for each of the following prediction algorithms:

Mean (M) The mean rating of the user
Pearson (P) The mean rating of other users weighted by their Pearson correlation to the user in question
Unit (U) The inverse resistance distance in the rating graph of unit resistances
Deep (D) The modified inverse resistance distance in the weighted rating graph

For each prediction method, we use linear regression (LR) to find an optimal affine function predicting the actual rating. We also use multiple linear regression (MLR) to predict ratings using different combinations of methods. For each combination, we calculate the mean squared error (MSE) and root mean squared error (RMSE) [10]. MSE and RMSE are *mean absolute error metrics* and the standard metrics to evaluate the predictive accuracy for recommender systems. For two vectors $x = (x_1, \ldots, x_n)^T$ and $y = (y_1, \ldots, y_n)^T$, the formula for MSE and RMSE is

$$\mathrm{RMSE(x,y)} = \sqrt{\mathrm{MSE(x,y)}} = \sqrt{\frac{\sum_{i=1}^{n}(x_i - y_i)^2}{n}} \tag{4}$$

---

[1] http://movielens.umn.edu/
[2] http://www.ieor.berkeley.edu/ goldberg/jester-data/

The test results are shown in Figure 10.

Corpus	MovieLens			Jester		
	MSE	RMSE	LR/MLR function	MSE	RMSE	LR/MLR function
M	0.265	0.514	0.0023 +0.9928M	0.240	0.490	0.01 +0.799M
P	0.315	0.560	0.276+0.0005P	0.280	0.537	0.0325+0.059P
U	0.307	0.554	0.1299+0.0027U	0.292	0.541	-0.149 +0.0073U
D	0.285	0.534	0.236+3.03D	0.261	0.510	0.044+1.38D
M+D	0.239	0.489	-0.024+0.952M+2.81D	0.204	0.452	0.024+0.828M+1.478D
M+P+D	0.239	0.489	-0.024+0.952M+0.004P+2.84D	0.203	0.451	0.024+0.828M+0.022P+1.417D

Fig. 10. Evaluation results. Numbers indicate the MSE (mean squared error) and the RMSE (root mean squared error).

We observe that the Pearson correlation has a much smaller predictive accuracy than the other predictions, and that the mean rating is the best single prediction method tested. As users tend to give rating only within a certain range, for example only rating movies they like using exclusively positive ratings.

The mean rating and deep similarity combined perform better than the mean rating alone, and adding the Pearson based prediction does not lower the error.

8 Conclusion and Future Work

In the rating graph given by a rating system, we have defined a measure of similarity between nodes. This measure is based on work from [1] extended by the support for negative ratings.

The modified inverse resistance distance was shown to predict ratings more accurately than a Pearson correlation based algorithm, and using it in combination with other basic prediction methods gives better results than any method for itself.

The following areas of research remain to be explored.

– Formulate a modified Y-Δ elimination such that the modified resistance distance is preserved. As we have seen, the trivial algorithm does not give the desired results. This line of research would allow an implementation to work on a graph-based representation of the problem.
– Compare and combine the modified inverse resistance distance with other prediction methods. Many other collaborative filtering algorithms exist [3] and could be used in combination with our approach. Since we have seen that a combination of predictions can lead to a better prediction, this line may be promising.
– Analyze the complexity of calculating the modified inverse resistance distance and use methods such as clustering to reduce the rating graph size. As done with other prediction algorithms, the rating graph may first be reduced using methods such as clustering to improve the runtime.
– The modified resistance distance can be calculated between any two nodes in the graph. Calculating it between two users or items leads to a similarity (or distance)

measure. This would be useful in recommendation systems and in social software when groups of similar users are to be detected.

References

1. Fouss, F., Pirotte, A., Saerens, M.: The application of new concepts of dissimilarities between nodes of a graph to collaborative filtering. ACM Trans. Math. Softw. (2003) Also TR-03-010 at www.cise.ufl.edu/tech-reports.
2. Billsus, D., Pazzani, M.J.: Learning collaborative information filters. In: Proc. 15th International Conf. on Machine Learning, Morgan Kaufmann, San Francisco, CA (1998) 46–54
3. Breese, J.S., Heckerman, D., Kadie, C.: Empirical analysis of predictive algorithms for collaborative filtering. In: Proc. of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence. (1998) 43–52
4. Mirza, B.J., Keller, B.J., Ramakrishnan, N.: Evaluating recommendation algorithms by graph analysis. CoRR **cs.IR/0104009** (2001)
5. Keller, B.J., mi Kim, S., Vemuri, N.S., Ramakrishnan, N., Perugini, S.: The good, bad and the indifferent: Explorations in recommender system health. San Diego, California, United States, http://www.grouplens.org/beyond2005/full/keller.pdf (2005)
6. Klein, D.J.: Resistance-distance sum rules. Croatica Chemica Acta **75**(2) (2002) 633–649
7. Qin, Z., Cheng, C.K.: Linear network reduction via Y-Δ-Transformation. technical report 2002-0706, University of California, San Diego, United States (2002)
8. George, A., Liu, W.H.: The evolution of the minimum degree ordering algorithm. SIAM Rev. **31**(1) (1989) 1–19
9. Heggernes, P., Eisenstat, S., Kumfert, G., Pothen, A.: The computational complexity of the minimum degree algorithm (2001)
10. Herlocker, J.L., Konstan, J.A., Terveen, L.G., Riedl, J.T.: Evaluating collaborative filtering recommender systems. ACM Trans. Inf. Syst. **22**(1) (2004) 5–53