Using Transitivity to Increase the Accuracy of Sample-Based Pearson Correlation Coefficients*

Taylor Phillips, Chris GauthierDickey, and Ramki Thurimella

University of Denver
Department of Computer Science
taylorphillips@du.edu,chrisg@cs.du.edu,ramki@cs.du.edu

Abstract. Pearson product-moment correlation coefficients are a well-practiced quantification of linear dependence seen across many fields. When calculating a sample-based correlation coefficient, the accuracy of the estimation is dependent on the quality and quantity of the sample. Like all statistical models, these correlation coefficients can suffer from overfitting, which results in the representation of random error instead of an underlying trend.
In this paper, we discuss how Pearson product-moment correlation coefficients can utilize information outside of the two items for which the correlation is being computed. By introducing a transitive relationship with one or more additional items that meet specified criterion, our Transitive Pearson product-moment correlation coefficient can significantly reduce the error, up to over 50%, of sparse, sample-based estimations. Finally, we demonstrate that if the data is too dense or too sparse, transitivity is detrimental in reducing the correlation estimation errors.

1 Introduction

Statistical models are used in the day-to-day lives of modern humans. Alleviating traffic congestion, predicting weather patterns, or investing in the stock market are all common examples of such models. When insufficient quantities of data are used by these models, they exhibit a phenomenon known as *overfitting*. This overfitting causes the models to display random error instead of an underlying trend, which in turn makes it difficult to utilize the results in a sensible fashion.

We propose an algorithm that reduces the effects of overfitting by using information in the data set other than that which the statistical model was built to utilize. For the purpose of discussion, we focus on a particular, ubiquitous example of a statistical model that is susceptible to overfitting known as the *Pearson product-moment correlation coefficient* (PMCC). This PMCC measures the correlation, or linear dependence, between two vectors, i and j, and relies solely on the intersection of those two vectors. Our proposed algorithm works by

* Published in the *Proceedings of the 12th International Conference on Data Warehousing and Knowledge Discovery (DaWaK 2010)*. Original version at http://www.springerlink.com

finding transitive neighbors, ks, such that the ks are the vectors in the data set most similar to i. These ks are then used to form estimates for i's relationship with j, allowing our algorithm to incorporate auxiliary information that is normally disregarded. Note that statistically PMCC does not exhibit transitivity: i.e., if X and Y are correlated and Y and Z are correlated, then X and Z are *not necessarily* correlated. Our goal, however, is to try to exploit those cases where the transitive relationship exists.

The existing approaches to alleviating the effects of overfitting do not address this issue directly. Instead, techniques *specific* to particular applications have been developed. Our work, which uses the idea of transitivity, could in theory be applied to improve estimations of many statistical models that use sparse data.

To quantify the performance of the algorithms presented in this paper, the Netflix Prize data set was used. This readily available data consists of approximately 100 million user-movie pairs. The results demonstrate that our Transitive Pearson product-moment correlation coefficient algorithm can reduce the error by up to 50% of the PMCC approximations in sparse data sets.

The notion of utilizing transitivity in statistical models to reduce the effects of sparse data is both abstract and powerful. The algorithm proposed in this paper is important because it is the first to demonstrate a significant reduction of error in sparse, sample-based PMCC estimations. PMCCs find uses in education, psychology, physics, mathematics, economics, and finance, all of which can suffer from overfitting and can subsequently benefit from the ideas presented in this paper. Further, this notion of neighbor transitivity used by our algorithm could be extended to reduce the error of other statistical models operating on sparse data.

2 Background

Collaborative filtering (CF) is the process by which users rate material in a collection or database of materials so that other users may use those ratings to help them select materials they are interested in [8]. The first opportunities for collaborative filtering came through the Internet where users were able to provide real time feedback on a product or service. Tivo [2] and Amazon were some of the first commercial entities to take advantage of it and an analysis of collaborative filtering in Amazon demonstrated a 20% increase in sales attributed to personalization through CF [11]. Adomavicius and Tuzhilin provide an extensive survey of collaborative filtering techniques [1].

Recently, the Netflix Prize offered $1,000,000 to anyone who could improve the accuracy of Netflix's proprietary movie recommendation algorithm by 10% [13,16,12,7,14]. Their algorithm was designed to recommend movies to customers based on how a specific customer rated his previously viewed movies. This same task of recommending movies could also be looked as the task of predicting a rating for an unseen movie, and then recommend movies with the highest predicted rating. Thus, the goal of the Netflix Prize was to improve the accuracy of the predictions of unseen movies.

Bell and Koren's work towards the Netflix Prize focused on nearest-neighbor models, latent-factor models, and combining those models together [3,5,10]. A synopsis of the research by Bell et al. on the Netflix Prize can be found in [6,4]. While being related to our own work, we are more interested in estimating the measures of similarity employed by these various algorithms.

The most widely used correlation coefficient in the Netflix Prize was the Pearson product-moment correlation coefficient (PMCC) [3,5,10,9,15], where it was used directly as a measure of similarity or to recommend an alternative. PMCCs are used as a measure of similarity in numerous applications of many different models, including matrix factorization and the nearest neighbor models.

Like any statistical model, the usefulness of the correlation coefficients is contingent on having sufficient data. While the Netflix dataset is large, the ratings are sparse and thus the challenge is to find relationships between users, movies and their ratings. Using the dataset, the PMCC is used to measure the correlation between two user's ratings. Thus, to compute the correlation coefficient for two users, an overlap in movies seen is needed to draw any conclusion about the relation between those two users. Our algorithm does not have this requirement and is able to make estimations of correlation coefficients when there is absolutely no data of this kind. This in turn allows models that make use of Pearson correlation coefficients to improve estimations of sample-based data and even make predictions that simply were not possible without this technique.

3 Motivation

Quantifying a relationship between users or items is an important component of collaborative filtering, with *similarity* being a measure of this relationship [3,5,10,9,15]. Similarity is then used to weight different opinions proportionally to the similarity of opinion between items. Determining the distribution of the weight from similarity is a specific focus in [5,10], which demonstrate how crucial similarity and weighting are when forming approximations from sparse data. There are multiple interpretations of similarity, but one commonly accepted method is correlation (linear or otherwise) [3,9,15].

3.1 Pearson Product-Moment Correlation Coefficient

The *Pearson product-moment correlation coefficient* (PMCC) is a measure of linear dependence between two vectors, i and j, in the range of [-1,1]. A PMCC of 1 indicates an exact positive correlation, -1 indicates an exact negative correlation, while 0 indicates there is no linear relationship. The formula for the PMCC of i and j can take many forms, one of which is shown in Eq. 1. The PMCC of i and j, dubbed r_{ij}, is based only on $i \cap j$ which are the points of data common between both variables. We will refer to $|i \cap j|$ or $|r_{ij}|$ as the *direct sample size* of i and j.

$$r_{ij} = \frac{\sum_x (i_x - \bar{i})(j_x - \bar{j})}{\sqrt{\sum_x (i_x - \bar{i})^2} \sqrt{\sum_x (j_x - \bar{j})^2}} \tag{1}$$

PMCCs are a standard for measuring linear dependence and thus, play a role in many fields ranging from math and statistics to social sciences and psychology. These correlations are, however, limited by the classic phrase that correlation does not not equal causation. This means that, for example, although temperature and humidity are negatively correlated, it does not imply that the increase in temperature caused the reduction in humidity. Correlations can still provide insight because they demonstrate that historical data indicates that there simply is a negative correlation, regardless of cause.

Other measures of similarity used in collaborative filtering include Euclidean distance and the Cosine similarity. Euclidean distance, defined as $d(i,j) = \sqrt{\sum_{x=1}^{n} i_x^2 - j_x^2}$, finds a natural usage when dealing with spatial proximities. The Cosine similarity finds the angle between vectors just as PMCCs find the slope between vectors and takes the form $cos(\theta) = \frac{i \cdot j}{||i||||j||}$.

3.2 Overfitting

PMCC can be limited in practice due to its susceptibility to *overfitting*. Overfitting, also referred to as inductive bias, is a symptom exhibited by statistical models that causes them to display random error instead of an underlying relationship. This means that a statistical model can indicate a relationship that is not true as a result of insufficient data. PMCCs are often used in sample-based scenarios which can have a small set of data that is not guaranteed to be representative of the theoretical, complete set of data. For example, if a statistical model relies on a single point of data, that point could be an outlier causing the model to predict incorrectly.

An alternative way of understanding overfitting is rooted in the law of large numbers. The law of large numbers states that the more data points that exist for a random variable, the more likely that data is to be representative of the expected value of that random variable. Rolling a single dice has six possible outcomes or values, all of which are equally likely. Because each outcome is equally likely, the expected value can be computed as the average of all outcomes, which for this example is $(1+2+3+4+5+6)/6 = 3.5$. If the dice is rolled only once yielding a one, there will be significant error if that single roll is assumed to be indicative of all possible rolls. Furthermore, statisticians are able to provide a confidence interval using the law of large numbers. That is, a sample size and interval may be specified such that the sample mean will fall within the specified interval the desired percentage of the time.

This idea may be applied analogously to the Netflix Prize where a given user's rating is the random variable in question. If we have only one rating, that rating is not necessarily representative of the long term opinions of that user. The law of large numbers states that the more ratings we have, the more likely that data will be representative of the long term. Thus, when dealing with an incomplete set of data, as in collaborative filtering, it is important to understand and account for this overfitting.

Various heuristics exist to curb overfitting and arrive at a more conservative estimate. This may be beneficial, or even necessary in some situations, but the

ability of these techniques to significantly improve the estimations is limited. Such heuristics can be as simple as skewing the original, overfitted value towards the mean of all values. In this paper, we propose an algorithm that minimizes the effects of overfitting of sample-based PMCCs by using information other than $i \cap j$. The myriad of applications of PMCCs can also suffer from overfitting and can subsequently benefit from the ideas presented in this paper.

4 Algorithms

In this section we describe two heuristics and our Transitive PMCC algorithm. For the following sections we assume there is a universe of vectors, for which a PMCC could be computed between any two vectors using Eq. 1. The algorithms will operate on some original PMCC, r_{ij}, and yield a new PMCC, r'_{ij}, that is intended to replace the r_{ij} for all subsequent applications.

4.1 Heuristics

The two heuristics presented in this section dampen the effects of overfitting by reducing the reliance on the data specific to r_{ij}. This is done by using a linear combination of the original r_{ij} and some given constant. The first heuristic algorithm, HeuristicA, takes two constants α and C where α is the linear combination weight given to r_{ij} and $1-\alpha$ is given to C. For our purposes we choose $C = 0$, indicating that the more weight C gets, the more it would transform r_{ij} to zero which, for PMCCs, means that there is no linearly dependent relationship between i and j. This choice of C curbs overfitting by skewing the actual r_{ij} towards this conservative value. HeuristicA is descibed formally in Eq. 2.

$$HeuristicA_{ij} = \alpha r_{ij} + (1-\alpha)C = \alpha r_{ij} \tag{2}$$

Note that α is not dependent on anything and thus, the linear combination weight given to r_{ij} is fixed. The problem with HeuristicA is that regardless of the direct sample size of r_{ij}, the linear combination weight remains fixed. Thus, an r_{ij} with a very large direct sample size would recieve exactly α weight just as an r_{ij} with a very small direct sample size. An improvement can be made by having the linear combination weight of r_{ij} be a function of the direct sample size of r_{ij}. This means that when there is a smaller direct sample size r_{ij} will get less weight, but as direct sample size increases r_{ij} gets more weight. This is useful because as direct sample size increases, the effects of overfitting ought to decrease and thus, the original value can be weighted more heavily.

HeuristicB uses $|r_{ij}|$ to arrive at a weight for r_{ij} that is more appropriate for the specific pair of i and j. It is described in Eq. 3 where β is the linear combination weight of r_{ij} and C is a chosen constant. As with HeuristicA, $C = 0$ was chosen so that the linear combination would be skewed towards 0, the equivalent of no relationship. For $\beta = 5$ with a direct sample size of 95, HeuristicB gives r_{ij} 95% of the linear combination weight and only 5% to no relationship.

If the direct sample size were only 5, r_{ij} would receive only 50% of the weight while no relationship would also get 50%. By determining a linear combination weight for r_{ij} from the direct sample size of r_{ij}, HeuristicB incorporates the conservative estimate with small amounts of data, but has little impact when there is larger amounts.

$$HeuristicB_{ij} = r_{i,j}\frac{|r_{ij}|}{|r_{ij}|+\beta} + (1 - \frac{|r_{ij}|}{|r_{ij}|+\beta})C = r_{i,j}\frac{|r_{ij}|}{|r_{ij}|+\beta} \tag{3}$$

4.2 Transitive PMCC

Our proposed Transitive PMCC algorithm (TPMCC) works to find information beyond $i \cap j$ to develop a stronger estimate for r'_{ij}. This extra information is rooted in the neighbors that are chosen to represent i's relationship with j. That is, the TPMCC algorithm takes the items most similar to i and uses their relationships with j to estimate i's relationship with j. To determine an ordering of neighbors for i by similarity, we use $abs(r_{ik})$, such that neighbors most similar will have a strong correlation. This strong linear dependence can be either positive or negative denoted by the absolute value.

The process of selecting a set of neighbors for a given pair, i and j, begins by examining all possible neighbor candidates, k. The candidates are then narrowed down, keeping only those whose $abs(r_{ik}) > \delta$ for which δ is some chosen constant. Additionally, we want to require some sufficient direct sample size on r_{ik} and r_{kj} so that we can have a degree of certainty that the neighbors themselves aren't suffering from overfitting. These constraints take the form of $|r_{ik}| \geq \gamma_{ik}$ and $|r_{kj}| \geq \gamma_{kj}$. We then take our final neighbor set N_{ij}, as the set of all ks that meet the previously stated criterion with δ, γ_{ik}, and γ_{kj}. The number of neighbors in N_{ij} will be referred to as the *transitive sample size*. The Transitive PMCC algorithm is then described in Eq. 4 where $w(i,j)$ is the weight of the actual r_{ij} and $w(i,j,k)$ is the weight of neighbor k.

$$TPMCC_{ij} = \frac{r_{ij}w(i,j) + \sum_{k \in N_{ij}} r_{kj}w(i,j,k)}{w(i,j) + \sum_{k \in N_{ij}} w(i,j,k)} \tag{4}$$

In order to examine all possible neighbor candidates, every unique PMCC must be computed. This step alone has an asymptotic complexity of $O(n^2)$ in running time where n is the number of vectors in the universe. The TPMCC algorithm then examines all $n-2$ neighbor candidates for each of the $O(n^2)$ unique PMCCs, making the asymptotic complexity of the Transitive PMCC algorithm $O(n^3)$. This is somewhat alleviated by being trivially executed in parallel, but the cubic complexity must be considered.

5 Experimental Methodology

The Netflix Prize data set was used to experimentally measure the performance of the heuristics and TPMCC algorithm presented in the previous section. This data set contains the rating history for 480,189 users and 17,770 movies with a total of 100,480,507 ratings. The movies data consists of a title, release year, and a unique identifier while the users consist of a unique identifier only. Lastly, the ratings data consists of a unique user identifier, a unique movie identifier, date of the rating, and a value of the rating ranging from one to five.

One hundred million ratings may appear substantial, but it only represents 1% of the total possible ratings. That is, if every user rated every movie then every possible rating would already be known, while in actuality 99% of those ratings are missing. This missing data complicates the use of PMCCs as they are based on only a subset of the possible data. Thus, the goal is to compute the PMCCs of the complete set of data using only a subset of the data. These sets could be thought of as a grading set and training set respectively. In our first set of experiments, we examine the effects of training the TPMCC algorithm on 50% of the data set (or 0.5% of the total possible ratings) to predict against the second half of the data. Subsequently, we will discuss how different amounts of training data influence the results.

For our purposes, the only points of data used were the rating's unique user identifier, the unique movie identifier, and the rating value. Using only one random half of the ratings data the PMCCs were computed for all pairs of movies. With 17,770 movies this results in 157,877,565 unique pairs of movies, each with their own PMCC. The PMCC for movies i and j in this set will be denoted $Original_{ij}$. Another set of PMCCs, $Final$, was computed using the entire set of ratings data and is used to grade the accuracy of the $Original$ PMCCs and the PMCCs created by the algorithms.

The $Original$ PMCCs will be used by the heuristics and TPMCC algorithm as input to provide new estimates for the $Final$ PMCCs. To quantify the error between any two sets of PMCCs, we use the root mean-squared error (RMSE). The formula for RMSE is shown in Eq. 5 where a and b are sets of PMCCs of size n.

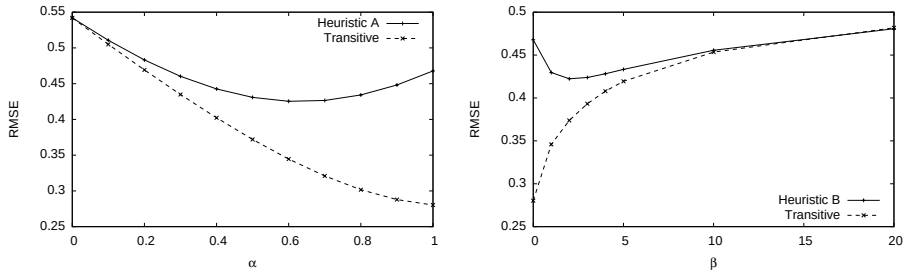
$$RMSE = \sqrt{\frac{1}{n}\sum_{x=1}^{n}(a_x - b_x)^2} \tag{5}$$

The resulting RMSE between $Original$ and $Final$ is 0.468. Theoretically, the worst possible RMSE could be 2.0. This would happen if, for example, the $Final$ PMCCs were all 1 and all of the $Original$ PMCCs were -1. However, given a distribution of data and predicting the mean yields much lower measures of error in practice. For example, the RMSE of absolutely no data, which is predicting 0 for every PMCC, yields an RMSE of 0.542. This means that using the $Original$ PMCCs computed using half of the data only reduced the error of predicting 0 for all PMCCs by 13.7%.

6 Results

6.1 Heuristics

Each heuristic was run using the *Original* PMCCs as input yielding two new sets of PMCCS, *HeuristicA* and *HeuristicB*. A plot of the RMSE of *HeuristicA* is shown in Figure 1a for different values of α. In this plot it is visible that the RMSE of *HeuristicA* is minimized for $\alpha = 0.6$, which reduced the RMSE to 0.425 - a 9.1% reduction of the RMSE of *Original*. The value of α that achieved the lowest RMSE is between 0 and 1, indicating that *Original* does suffer from overfitting and benefits from the HeuristicA algorithm. If $\alpha = 0$ or $\alpha = 1$ yielded the least RMSE, it would mean that predicting 0 for all PMCCs was best or using the unmodified *HeuristicA* was best, respectively. When $\alpha = 0.6$, HeuristicA is going to scale the *Original* PMCCs down to 60% of the linear combination weight and give 40% to 0. The contrast with the effects of HeuristicA on the PMCCs produced by the TPMCC algorithm as also shown in Figure 1a will be discussed in the following section.



(a) HeuristicA (b) HeuristicB

Fig. 1: *RMSE of HeuristicA vs. α and HeuristicB vs. β:* HeuristicA exhibits the lowest RMSE for *Original* when α is 0.6 for a 9.1% improvement of the RMSE of *Original* while HeuristicB exhibits the lowest RMSE with *Original* when β is 2, yielding a 9.8% improvement of the RMSE over *Original*.

The RMSE of the PMCCs of the HeuristicB algorithm were computed for various values of β and displayed in Figure 1b. Note that choosing $\beta = 0$ results in no change to *Original* and $\beta = \infty$ would result in a prediction of 0 for all PMCCs. The RMSE for *HeuristicB* was minimized using $\beta = 2$, which achieved a total reduction of RMSE of nearly 9.8% over *Original*. Like HeuristicA, this demonstrates that HeuristicB does reduce the RMSE of the PMCCs indicating that *Original* does suffer from overfitting. This value of β means that PMCCs with a direct sample size of 2 were reduced to 50% of their value, while PMCCs with a direct sample size of 20 were reduced to only 90.1% of their original value. The contrast with the effects of HeuristicB on the PMCCs produced by the

TPMCC algorithm as also shown in Figure 1b will be discussed in the following section.

6.2 Transitive PMCC

Multiple sets of PMCCs were computed with the proposed TPMCC algorithm using *Original* as input. The different sets were computed with different constraints on the neighbor sets. We chose a fixed $\delta = 0.9$ and $\gamma_{kj} = 1$, but used multiple values of γ_{ik} ranging from 3, 6, 12, and 24. This means that the neighbors for the $Orginal_{ij}$ were limited to ks such that $abs(Original_{ik}) \geq 0.9$, the direct sample size of $Original_{kj}$ is greater than zero and the direct sample size of $Original_{ik}$ ranged from greater than or equal to 3, 6, 12, and 24. Our implementation was in Java and the computation was performed in parallel on four machines. The machines had 4GB of RAM, 2.13GHz Intel Core 2 CPU and were running Debian GNU/linux 2.6.18. Depending on the value for γ_{ik} (which determined the size of the neighbor sets), the entire operation would take five to eight hours. In contrast, a standard PMCC calculation could be done on a single machine in less than an hour.

The RMSEs for the different values of γ_{ik} are shown in Figure 2. The second axis of the figure displays the average number of transitive and direct neighbors of i and k for each γ_{ik}. Note that with $\gamma_{ik} = 24$ it was difficult to even find a large number of direct neighbors and thus, didn't have a significant impact on the data. Both the improvements from $\gamma_{ik} = 24$ to $\gamma_{ik} = 12$ and $\gamma_{ik} = 12$ to $\gamma_{ik} = 6$ were substantial, while the change form $\gamma_{ik} = 6$ to $\gamma_{ik} = 3$ had little impact. This shows that direct sample sizes like 6 and 12 held a strong balance between attainability and usefulness. Neighbors that only have a very small direct sample size are less reliable because such a small direct sample size could easily misrepresent the complete set of data, however, they were still able to make a positive contribution to reducing the overall RMSE. The TPMCC algorithm is minimized for $\gamma_{ik} = 3$ with nearly 1300 transitive neighbors, which reduces the RMSE of *Original* to 0.28, a 40.1% reduction in RMSE. This set of PMCCs, denoted $Transitive$, will be used in subsequent comparisons to other sets of PMCCs.

Looking back at Figures 1a and 1b, both plots also display the results of each heuristic algorithm on the PMCCs produced by TPMCC. In these figures $Transitive$ is minimized by the heuristic algorithms when they don't effect them at all - namely $\alpha = 1$ and $\beta = 0$ for HeuristicA and HeuristicB respectively. As discussed in the above subsection, these values of α and β have no effect on $Transitive$, and further, the RMSE gets progressively worse as the heuristics make a larger impact. This is directly indicative that $Transitive$, unlike *Original*, already accounts for overfitting and is only made worse by the heuristics.

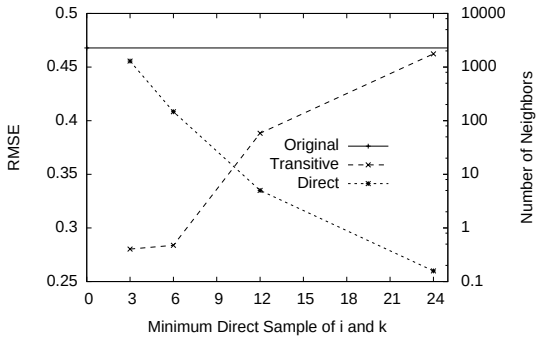


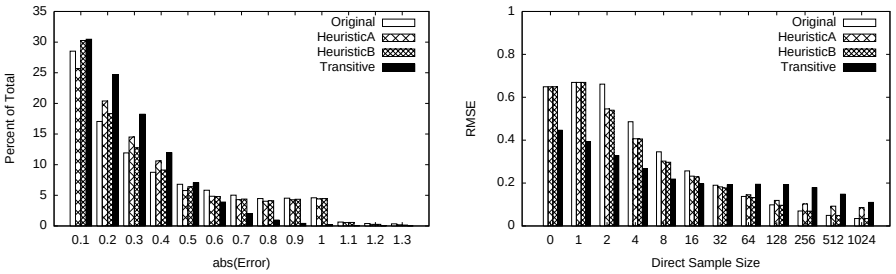
Fig. 2: RMSE of TPMCC with *Transitive* vs. γ_{ik}. The RMSE for *Original* is also displayed for comparison. The secondary Y axis is the mean number of transitive neighbors for the *Neighbors* plot. Note that *Transitive* exhibits the lowest RMSE when γ_{ik} is 3, computed with nearly 1300 neighbors, yielding a 40.1% improvement over *Original*.

6.3 Error Distributions of PMCC Estimations

The distribution of error from each algorithm's PMCCs, including *Original*, are shown in Figure 3a. The plot was built by computing the absolute value of the error and counting the frequency of errors falling into each bucket. The buckets have a lower and upper threshold, all of which were chosen to have width 0.1 and range from 0 to 2. A particular error value falls into the first bucket for which the error is less than that bucket's upper threshold. We will refer to the first bucket, containing values ranging from 0 to 0.1, as the "0.1 bucket" and all subsequent buckets will be denoted by their upper threshold.

In Figure 3a, *Original* has the largest error of any other set of PMCCs. The *Transitive* PMCCs contain the most values in the 0.1 bucket with over 30% of all PMCCs falling into this category. Both heuristic PMCC sets are close behind, while *Original* has only 25%. In the next two buckets, *Original* and both heuristics differ slightly, but TPMCC has about 5% more. In addition, *Transitive* is the only set of PMCCs to have any significant effect on buckets 0.7 to 1, which each contain roughly 5% of all other sets of PMCCs. *Transitive* has much less, emphasizing the fact that it has much fewer high-error PMCCs. These buckets are likely populated by PMCCs that have a very small direct sample size which results in overfitting and high error. TPMCC's performance in this situation is indicative that the it is doing more than curing the symptoms of overfitting, but actually using the extra information to improve estimations.

To further examine the PMCCs and understand the implications of Figure 3a, a second distribution was made to show the RMSE for different direct sample sizes of i and j in Figure 3b. Like Figure 3a, this distribution was sampled using thresholds and each bucket is denoted by its upper threshold, where the first bucket contains only those PMCCs who had a direct sample size of zero. The



(a) Distribution of Absolute Error (b) Distribution of RMSE

Fig. 3: *Distributions of the absolute value of error and RMSEs:* In (a), the distribution of the absolute value of the error with thresholds of width 0.1 where the X axis represents upper threshold is shown. Note that *Transitive* has the most values in buckets 0.1-0.5 while the others have more in the high-error buckets. In (b), the distribution of RMSE by direct sample sizes of PMCCs where the X axis represents upper threshold (inclusive). Note that *Transitive* exhibits significantly lower RMSEs for direct sample sizes less than or equal to 8. After this point *Transitive* has higher RMSEs demonstrating that the transitive data becomes less valuable as direct sample size increases.

remaining buckets have exponential widths ranging from the previous buckets upper threshold (exclusive) to its own upper threshold (inclusive).

The PMCCs for all algorithms, excluding *Transitive*, are identical for the first two buckets, 0 and 1, as they had no data on which to operate and therefore predicted 0. The TPMCC algorithm was able to produce PMCCs that reduced this error in bucket 0 by over 31.2%. For bucket 1, *Transitive* further improves and demonstrates its ability to operate with little direct data and reduces the error of all other algorithms by 41.4%. *Original* doesn't improve much in bucket 2, but both heuristics show a drastic change and reduce the RMSE of *Original* by 17%. The *Transitive* continues to improve and reduces the RMSE of *Original* for the bucket 2 by 50.4%. These contrasting results demonstrate the ability of the TPMCC algorithm to extract indirect information from transitive neighbors and improve the accuracy of predictions with limited amounts of data.

The *Transitive* continues to outperform all other algorithms by a similarly significant margins up to bucket 8. For buckets larger than 16, a new trend develops and *Transitive* begins to have a larger RMSE than the other algorithms. This interesting behavior implies that there exists a direct sample size at which point enough direct information renders the transitive neighbor information detrimental. This is somewhat intuitive as the larger the direct sample size that is available, the more trust that can be placed on the subsequent results. Thus, when the results are sufficiently trusted, the *Transitive* uses less accurate and indirect information from transitive neighbors that actually increases the RMSE of the PMCC estimations.

To gain insight as to how significantly Figure 3b will impact the overall RMSE, a third distribution was made. This distribution is shown in Figure 4 and displays the percent of all PMCCs to fall in each of the buckets used in Figure 3b. It shows that roughly 7% of all PMCCs have a direct sample size of 0. The thresholds with direct sample sizes from 0 to 8 account for 67.9% of all data and *Transitive* was able to reduce the RMSE by 42.7%. In addition, the thresholds where *Transitive* is detrimental, direct sample sizes with thresholds 64 and greater, all combine to make up only 12.9% of all unique pairs.

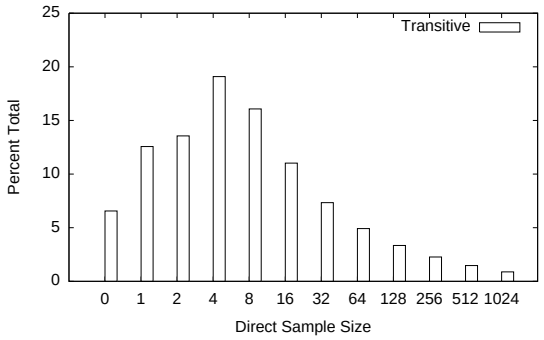


Fig. 4: Distribution of all PMCCs by direct sample sizes where X axis represents upper threshold. Note that samples sizes 0 through 8, where TPMCC performs well, account for 67.9% of all PMCCs. The TPMCC has a negative impact on buckets with direct sample sizes of 64 and greater, which combine to only 12.9%.

6.4 Data Density

As discussed in Section 5, the Netflix Prize data set has approximately a 1% density, indicating that in the results presented above the algorithms operated on only 0.5% data density since we used half of the data set. The following analysis addresses how the different algorithms perform as the amount of data is reduced.

The plot in Figure 5 shows a RMSE of different sets of PMCCs as the amount of data is varied. $OrigHeuristicA$ and $OrigHeuristicB$ are the PMCCs resulting from using *Original* with each heuristic. *Transitive* is the results from TPMCC algorithm while $TransHeuristicA$ and $TransHeuristicB$ are from the heuristics operating on *Transitive*. With 0% of the data, all algorithms produce the same set of PMCCs which amounts to predicting 0 for all PMCCs. The 2% sample points, or 0.02% data density, show little improvement because the data is still too sparse to support sufficient intersection between movies.

To confirm that 0.02% data density was too low, Figure 6 displays the mean number of transitive neighbors found by TPMCC and the number of direct

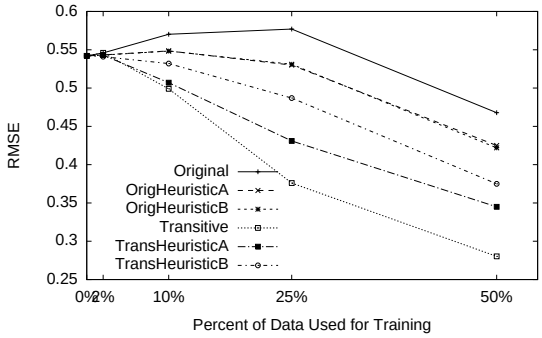


Fig. 5: RMSE vs. percent of Netflix Prize data used, *Transitive* and both heuristics for each. Note that *Original* actually increases RMSE as amount of data increases until 50% data, while *Transitive* is able to make increasingly notable improvements starting with 10% data.

neighbors for normal PMCCs. For the 2% predictions, it is clear that neither transitive nor direct neighbors exist in usable quantities.

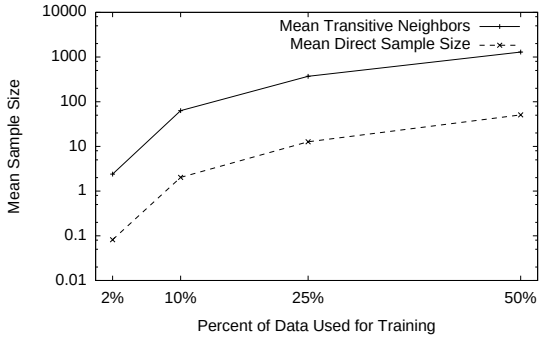


Fig. 6: Mean direct sample size and mean transitive sample size vs. percent of Netflix Prize data used. Note that neither reach usable quantities for 2% data density, but at 10% the mean number of transitive neighbors reaches 50, while mean number of direct neighbors only reaches just over 2. At 50% data there are nearly 1300 and 50 mean transitive and direct neighbors respectively.

The next point at 10% (0.1% data density) shows an interesting trend. With nearly 100 transitive neighbors, *Transitive* reduces the RMSE of *Original*, by 12.5%. The heuristics for *Transitive* fail to make additional improvement. *Original* and its heuristics actually perform slightly worse than predicting 0 for all PMCCs. This demonstrates that such sparse data causes overfitting, and in this case, can actually be improved by not utilizing the data at all.

At 25% (0.25% data density) *Original* is still being out performed by predicting 0 for all PMCCs. Its heuristics do slightly better, but Transitive is able to achieve a 34.8% reduction in RMSE over *Original*. Back to 50% of the Netflix Prize data, *Original* makes a drastic improvement with a mean intersection size of 51. *Transitive* continues to improve reaching a 40.1% reduction over *Original*. Note that the heuristics operating on *Transitive* are always only made worse, demonstrating that the TPMCC algorithm has already reduced overfitting beyond the aid of those heuristics.

It is interesting to note that the Transitive PMCC algorithm is able to begin reducing RMSE with only 100 transitive neighbors as shown in the 0.1% data density point in the plots above. However, with 0.5% data density, it further benefits from over 1000 neighbors. This means that the Transitive algorithm benefits from being in a *wide* data set, or a data set that has lots of users and movies in the case of the Netflix data set. If there were only 10 movies, it would be very difficult for the Transitive algorithm to find a sufficient number of neighbors. With the Netflix Prize data, almost 20,000 movies along with half a million users exist, given plenty of opportunities to find different neighbors, transitive or not.

7 Conclusions and Future Work

The proposed nearest neighbor PMCC algorithm increases the accuracy of PMCCs estimations when dealing with sparse, sample-based data. In such sample-based data, statistical models can suffer from the lack of data and represent random error instead of underlying trends in a phenomenon known as overfitting. The results of the experiments with the Netflix Prize data demonstrate that the proposed heuristics and TPMCC algorithm are able to reduce the error in such PMCC estimations.

The PMCCs computed from the random test set reduced the error of predicting 0 for all PMCCs by only 13.7%. The heuristics reduced the error of the test set PMCCs by up to 9.8%, while our TPMCC algorithm, which took advantage of transitive relationships, was able to achieve a 40.1% reduction. For Pearson estimates with direct sample sizes of two, which account for 13.6% of the population, the TPMCC reduced the error by over 50%. Lastly, the TPMCC algorithm is able to provide comparable improvements with reduced amounts of data. This reduction in error of PMCCs will strengthen the variety of applications in which they are applied and allow statistical models to be utilized in situations where they otherwise could not. Furthermore, the abstract notion of gathering information from transitive neighbors is likely to have a positive effect in new applications.

For future work, we plan on exploring how measures of similarity other than PMCC, like the Jaccard index, Euclidean distance, and Spearman rank coefficient, could be improved by discovering transitive relationships in the data sets. The TPMCC's temporal computational complexity is $O(n^3)$, requiring an $O(n)$ operation for each of the $O(n^2)$ unique PMCCs. This running time could be

reduced to $O(kn^2) = O(n^2)$ by selecting some well- chosen subset of size k to represent all possible neighbor candidates. Furthermore, if the TPMCC algorithm was computed on a subset of c PMCCs, those that are likely to benefit the most (e.g. those with a very small sample size), it could be reduced to $O(ckn) = O(n)$ which could make it much more pragmatic in real life situations.

References

1. Adomavicius, G., Tuzhilin, A.: Towards the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge and Data Engineering 17, 634–749 (2005)
2. Ali, K., van Stam, W.: Tivo: Making show recommendations using a distributed collaborative filtering architecture. In: Proceedings of the 10th ACM International Conference on Knowledge Discovery and Data Mining. pp. 394–401 (2004)
3. Bell, R., Koren, Y.: Improved neighborhood-based collaborative filtering. In: International Conference on Knowledge Discovery and Data Mining (2007)
4. Bell, R., Koren, Y.: Lessons from the netflix prize challenge. SIGKDD Explorations 9, 75–79 (2007)
5. Bell, R., Koren, Y.: Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In: IEEE International Conference on Data Mining (ICDM'07). pp. 43–52 (2007)
6. Bell, R., Koren, Y., Volinsky, C.: The bellkor solution to the netflix prize. Tech. rep., AT&T Labs (2007)
7. Buskirk, E.V.: Winning teams join to qualify for $1 million netflix prize. Wired Magazine (2009)
8. Goldberg, D., Nichols, D., Oki, B.M., Terry, D.: Using collaborative filtering to weave an information tapestry. Communications of the ACM 35(12), 61–71 (December 1992)
9. Hong, T., Tsamis, D.: Use of knn for the netflix prize. Tech. rep., Stanford University (2006)
10. Koren, Y.: Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: International Conference on Knowledge Discovery and Data Mining (2008)
11. Linden, G., Smith, B., York, J.: Amazon.com recommendations: Item-to-item collaborative filtering. IEEE Internet Computing 7(1), 76–80 (Jan/Feb 2003)
12. Lohr, S.: Netflix Competitors Learn the Power of Teamwork. NY Times (2009)
13. Netflix: The Netflix Prize. `http://www.netflixprize.com`
14. Newitz, A.: Movie Tips From Your Robot Overlords. Washington Post (2009)
15. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Item-based collaborative filtering recommendation algorithms. In: Proc. 10th International Conference on the World Wide Web. pp. 285–295 (2001)
16. Thompson, C.: Netflix challenge to hackers: Improve our service and win big. NY Times (2007)