The Netflix Prize: Exploration of High Density in Recommender Systems in a Large Dataset

Timothy Ozimek
North Carolina State University
teozimek@ncsu.edu

1 Background

The movie service, Netflix, offers customers the choice of watching several thousand movies on demand. While the service now offers a highly popular online, on-demand, streaming delivery method, in the past, operations were run through a mail DVD system. A user of the service could request a certain number of movies at any given time. After watching and returning the DVDs, a customer could receive new movies as quickly as the mail service could deliver them. Along with the movies, Netflix provided a list of recommended movies based on a user's viewing preferences. A viewer can voluntarily rate movies they've seen with the understanding that their rating would guide Netflix in suggesting new movies. The more ratings they gave, the better the recommendations became.

To provide this service, Netflix wrote the software tool, *Cinematch* [1]. This tool used a combination of machine learning and data mining algorithms to perform its duties. It uses the aptly named recommender systems, which have received serious study over the years. The intuition behind these systems is based on how word-of-mouth conversations spread news of a product. These conversations occur between friends and acquaintances who tend to share common opinions over a variety of topics. Likes and dislikes accordingly tend to correlate.

The field of data mining and machine learning has observed this relationship and formalized the task in a recommender system. Such systems tend to employ the technique of collaborative filtering. In short, this method scours data, looking for similarities in rating patterns, much like word-of-mouth works among similar friends, but at a far larger scale. The result of this similarity search will filter a group of similar users. This filtered group provides numerical information to compute a recommendation[2].

A variety of algorithms have been adopted. Two stand large: item-based collaborative filtering (IBCF) and user-based collaborative filtering (UBCF). The names of each imply their method of search. IBCF finds similarity of items, in this case movies, and makes recommendations based on the ratings of a target user. Likewise UBCF establishes similarity through the intuitive word-of-mouth approach by comparing similar user ratings[3].

To stimulate research in improving their recommender system, Netflix offered the Netflix prize in 2006. The objective seemed tantalizingly simple: be the first team to improve recommendations by 10% over *Cinematch* and receive $1 million. Without delving too far into details, contestants were given a dataset of over 100 million user ratings of 17700 movies. Against this dataset algorithms could be devised, tested, and submitted to Netflix. Netflix would run the submission against a hidden testset. A simple root-mean-square-error would be computed per tested rating to arrive at a final score. This score was the metric compared against *Cinematch* to determine the winner[1].

The prize was captured in September 2009 by the team "BellKor's Pragmatic Chaos", a group of professional statisticians and machine learning experts. Their methods were sophisticated, entirely new, and were comprised of 107 predictor algorithms run as an ensemble. At their core, however, they were recommender systems that had advanced ways of producing better similarity metrics[3].

The objective of this paper is less ambitious than the Netflix Prize. The combined efforts of the winning team included well over 2000 hours of work for their preliminary solution where even their simplest algorithms took 45 minutes to run. Several searches spanned over many hours[4]. The scope is beyond what can be provided in less than a semester's time.

Instead this paper will examine a high density subset of the dataset. Initial exploration of dataset included the use of the *recommenderlab* tool suite, a set of tools and algorithms in R "which provides the infrastructure to develop and test recommender algorithms for rating data and 0-1 data in a unified framework"[5]. While the tool was useful for exploring small datasets, it did not scale with high memory / density matrices and therefore was abandoned.

High density simply means that the user-item ratings matrix contains enough entries to fill the matrix past a pre-determined percentage. The study [3] explores performance for different collaborative filtering algorithms especially towards the lower end of the density spectrum. It notes that depending on density, different approaches should be taken to get collaborative filtering to work.

The objective of this paper is to explore the high density region (> 20 %) with user-based collaborative filtering, item-based collaborative filtering, different methods of similarity comparison, finding 'k' nearest neighbor parameters, comparing the effects of normalization, finding if UBCF or IBCF trends better, and establishing a relationship between accuracy and various levels of density in this high density region.

2 Method

The approach to the project is essentially a survey of different recommender systems. The kernel of these approaches are the two main collaborative filtering techniques: user and item based. Two main programs run the algorithms of each, while several others will construct appropriate input datasets. Parameters for normalization, similarity, dataset, and nearest neighbors will be supplied to each. Below is a discussion of each method.

2.1 UBCF

This technique is given a set of test users that request ratings for a randomly selected movie of which they have rated. Against this training set, all users who have rated the test movie are extracted. In this subset of users, a similarity comparison is done against the test user. A sorted list is returned. From the most similar users in this list, as set by a parameter k, a rating can be made by averaging the results of their scores. The search is expensive because there is no easy way of visiting the entire user space, especially with a dense subset. Because this technique compares user-to-user directly, it receives the UBCF designation.

2.2 IBCF

IBCF seeks items that are most similar to each other. In most traditional uses of IBCF, a list of most similar items are returned based on a similarity score. The algorithm has been modified to instead return a user's estimated rating from a similarity matrix.

The first step is to construct a user / movie ratings matrix where all ratings of the training set are filled accordingly. Then, systematically, each item's ratings are compared to all other items. A similarity computation is performed that yields a similarity number. This number is stored in a user / movie similarity matrix. Note that this matrix is triangular and will be roughly half the size of the ratings matrix.

When a test user requests a rating, the test movie's row index is used. Along this row are returned the similarity values for all of the test user's known ratings. A weighted sum of the user's own ratings is used via the formula to construct the final result:

$$\text{Rating} = \frac{\sum_{i=1}^{k}(r_i * sim_i)}{\sum_{i=1}^{k}(sim_i)}$$

Another weighing approach is to use the sum squared average. The highest similarity is always 1 therefore the lower the similarity the more the squared rating is penalized:

$$\text{Rating} = \frac{\sum_{i=1}^{k}(r_i * sim_i^2)}{\sum_{i=1}^{k}(sim_i^2)}$$

2.3 Similarity Metrics: Cosine and Pearson

Similarity metrics in this paper follow two main formulations. Cosine distance measures similarity via the formula:

$$cos(\theta) = similarity(A, B) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{N} A_i B_i}{\sqrt{\sum_{i=1}^{N} A_i^2}\sqrt{\sum_{i=1}^{N} B_i^2}}$$

Pearson similarity is computed as:

$$r = similarity(A, B) = \frac{\sum_{i=1}^{N}(A_i - \bar{A})(B_i - \bar{B})}{\sqrt{\sum_{i=1}^{N}(A_i - \bar{A})^2}\sqrt{\sum_{i=1}^{N}(B_i - \bar{B})^2}}$$

Both forms will be examined across UBCF and IBCF. The Pearson is expensive because it requires two passes of the data, one for the average, and the second for the similarity. However, this cost may be justified because of an inherent tendency to normalize.

2.4 Normalization

Users can be optimists or pessimists on all their ratings or for portions of their ratings. To see the effect of this bias, similarity measurements will be conducted under normalized and non-normalized data. In UBCF two main biases will be normalized. The first will be for computation of similarity. The second will be on the returned score to the user, to shift back into the biased domain.

For IBCF, the similarity matrix is constructed with normalized ratings. Because only a similarity is returned, a test user's bias need not be computed.

The normalization formula used is a shifted bias from the average rating a user gave on all movies that they rated:

$$Rating_i = Rating_{ui} - \overline{Rating_u}$$

2.5 Blending of Global Data for Cold Starts or Sparse Matches

In cases where similarity yields undefined results, for example, floating point overflow or underflow, the average rating for a movie is used instead. Care is taken to check for such conditions.

2.6 Other Approaches Considered

There are many other areas to explore with recommender systems. Some mentioned in [3], [4], [6], and [7] include Restricted Boltzmann Machines (RBM), Matrix Factorization / Single Value Decomposition (SVD), and Slope-One estimation. Even more exotic approaches described in [2] include simulated annealing. Indeed, the space of applications is very large and continually growing. Computation and limited research time prevented exploration into these areas. [1] indicates that the Netflix Prize intermediate solution took 2000 hours and involved 107 algorithms for RBM and SVD. This amount of time exceeds the amount alloted for the assignment. The Slope-One estimation was more appropriate for linear regressions but not with the implemented system in this paper. Finally, only the high density region of the dataset was explored. The paper [7] notes that different approaches might be considered depending on the data density with higher density producing more

reliable results. The results of this project might show an upper bound expected for these types of algorithms.

3 Plan and Experiment

3.1 Dataset

The dataset is comprised of over 100 million ratings of 17,700 movies from 480,189 users. 17,700 text files, one per movie, contain a user id, rating, and date per line in ASCII comma-separated format. A movie ID value was given on the first line of the file as an integer number. A separate file gives the movie title for a given ID. User IDs were arbitrary integers assigned by Netflix that stripped away personally identifying information. The same ID is used by a user for all movies that they rate. Ratings are ordinal integer values from 1 to 5, with 1 indicating the worst rating and 5 the best. Dates are also given in month-day-year format.

Of the above data, only the movie ID, user ID, and ratings are used in this paper. No testing sets are provided in the original data. The test sets are generated from the given data through a process described in the experimental section of the paper. Auxilliary programs convert all relevant data into one master file in binary format to conserve on space and load times. Subsampled data is also in this format.

3.2 Hypotheses / Questions

The examination of the dataset will attempt to answer the follow questions.

- This paper posits that the ideal k parameter in the k nearest neighbors algorithm will vary significantly between IBCF and UBCF and will need to be greater than at least 5 in either case. Both CF algorithms can return an arbitrary number of similar users. Cosine and Pearson similarity return scores in the range of -1 to 1. Values closer to 1 indicate higher similarity. Finding the ideal k parameter for k nearest neighbors will govern the overall performance of the algorithm.

- Between the similarity metrics, this paper expects Pearson similarity to outperform Cosine similarity in all regards. Pearson similarity is a correlation metric that captures global effects by using the average value of each vector. This can be viewed as a built-in normalization effect. Furthermore, Pearson against normalized and untreated data will give similar results.

- Normalization will offer some benefit to both algorithms over untreated data. However, the effect should be relatively small. The ratings space only occupies 5 ordinal values; there is not much space where normalization values can settle.

- High density data will yield superb results for UBCF and IBCF with RMSE results on par with the Netflix Prize winners results (0.8650 RMSE) on the overall dataset. These are the best conditions for such data. However, a cross-over point will be observed where average RMSE will beat these metrics. A simple test against a low density sample showed that RMSE scored 1.05 while the other methods ranged from 1.20 to 1.32 RMSE. A cross-over point might occur in the 30 % density region where the results could deteriorate.

- User based collaborative filtering will work better than item based collaborative filtering. This is just a guess. It will be interesting to see which method produces better results.

- The relationship for a fixed number of movies and users in a subsample may reveal where each trend will hit their respective cross-over points as compared to the baseline metric. Essentially, this question expects an inverse linear relationship between RMSE per density.

3.3 Experimental Design

The overall experiment will follow the guidelines for evaluation as set by the Netflix Prize. A test set (not provided in the original dataset) will query the recommender system which will give a decimal rating for the user based on that user's prior movie ratings. Error measurements are computed by the root-mean-square-error formula:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}{N}}$$

Unlike linear error methods, such as MAE, RMSE penalizes errors more severely. Accuracy is therefore, at a premium.

Subsamples are generated from the highest top X movies and top Y users in terms of raw number of ratings count. The subsamples will fill a 5 x 5 matrix of varying X and Y counts.

From each subsample, 10-fold cross validation is performed. For the testing portion of a fold, a test subject will randomly have a rating stripped from the test set. The rating will be stored separately for later RMSE computation against that user's predicted value.

Randomness will be controlled through C++11's uniform random number generator with seedings recorded for reproducibility.

The baseline metric will always be the average rating for a movie over the entire training set. This baseline is used by all citations that analyze Netflix Prize recommender system algorithms.

The first k nearest-neighbor experiment will determine the k parameter to be used in all subsequent experiments. All combinations similarity metric, CF algorithm, and normalization will be used to find which best k value to use per setting. The results will be computed against the (Movie = 1500 x User = 500) subsample with k being the independent variable.

The remaining experiments will use the entire 5 x 5 set of high density subsamples and observe the k parameter discovered above. Comparisons are between: Cosine and Pearson . Normalize and Untreated Data, and UBCF versus IBCF.

For uncovering user and movie dependencies, either parameter will be fixed while the other varies.

4 Results

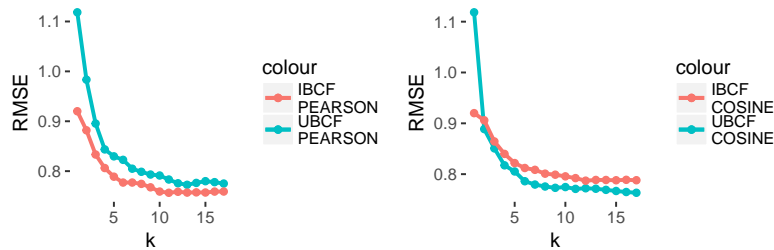### 4.1 K Nearest Neighbor Parameter Search

Figure 1: K nearest neighbor parameter search

At lower values, k is susceptible to one or two dominating neighbors who may overemphasize certain traits at the expense of others. In effect, they become outliers.

At the higher extreme, the k parameter saturates RMSE scores. This is because as more neighbors are added the less they becomed differentiated from each other. Thus, the differences in similarity start offering diminishing returns. Consider that if all elements are returned into the dataset that the estimated rating would become the overall average rating. This implies the the k curve will eventually worsen over time rather than incrementally improve.

The above graphs are examples of the overall trend in k values. An adequate location for k appears in the 8 to 12 region. All remaining experiments set the value to 10.
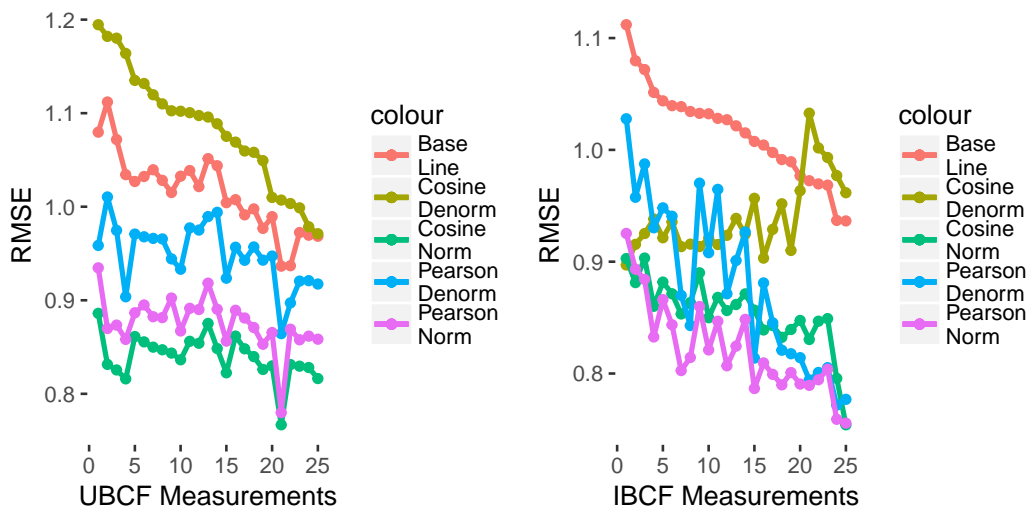
Figure 2: Collaborative filtering normalization and similarity comparison

4.2 UBCF and IBCF Similarity and Normalization Comparisons

Over all the combinations of the 25 measurements from the 5 x 5 density matrix, overall similarity and normalization techniques can be compared to the baseline. In the UBCF case, normalization squarely outperforms its untreated counterpart by significant margins. The Cosine Denormalized metric is worse than even the baseline. In the IBCF the trend is similar. Overall normalization offers anywhere from 10% to 20% RMSE improvement over the baseline. This relatively simple technique demonstrates that with even 5 ordinal ratings, enough of a consistent bias exists on a per user basis to justify aligning all data to a common unbiased axis. This result was signficant and surprising.

The performance of Pearson versus Cosine Normalized is different for UBCF and IBCF with Cosine prevailing in the former and Pearson in the latter. The expectation was that Pearson would work best in all conditions. While the improvment is modest it is consistent throughout the experiment.

This data also answer another question. Raw high density subsamples match or exceed even the Netflix Prize results. This is fairly unsurprising; few gaps exist in the dataset.
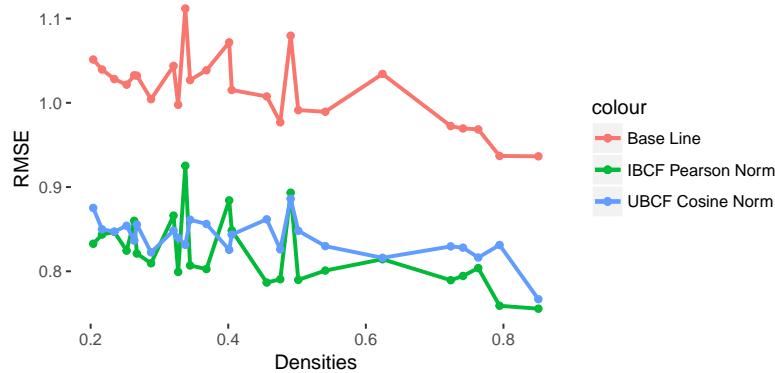
4.3 Overall Density vs. RMSE

Figure 3: Density vs. RMSE for highest performing metrics

6

324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377

For densities down to 20%, the top collaborative filtering methods continue to signicantly outperform the baseline metric. While there appears to be a general trend of worsening results towards the lower densities, the baseline also tracks in this direction. This indicates that the crossover point could be much lower than 20%, possible in low single digit density regions. This is a testament to the success of these algorithms, especially where data is well represented.
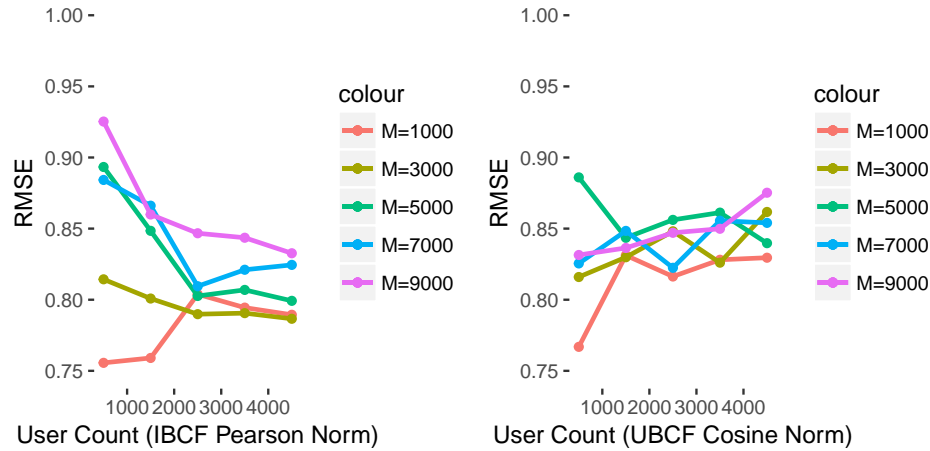
4.4 IBCF vs. UBCF Results

Figure 4: RMSE for fixed movie count vs. user count of IBCF and UBCF

These plots show converging results after movie and user count exceed a certain portion of the subsamples. The trend would indicate that the IBCF performs better than UBCF for the dataset subsamples. There is a wide variability at low user count settings for both CF metrics, suggesting not enough user information is available when a subsample is low as compared to the overall dataset. This indicates that future experiments should possibly consider some threshold percentage of both data dimensions before results start to stabilize. Where the results start to converge, IBCF appears to outperform UBCF, marginally so. A possible explanation is that the number of movies is outstripped by the total user count, indicating more useful information is stored in the item-item matrix.

4.5 Miscellaneous Results

In cases where data points were missing (e.g. the k-nn metric was not met), the average rating was used instead. This proved to have insignificant effect on the data.

For weighted similarities in the item-item matrix, a linear and squared rating appearing to only have a 0.0001 overall RMSE improvement with the squared result slightly favored.

5 Conclusions

For high density datasets ($> 20\%$) both UBCF and IBCF are highly successful in predicting user ratings over the baseline. The success of these algorithms probably extends much lower, likely into single digit densities. This area would be interesting to explore, but time constraints prevent investigation into this area. Normalization offers outstanding results, in excess of 20% improvement in RMSE, and should be a crucial part in all collaborative filtering algorithms. Similarity scoring varies between UBCF versus IBCF and should be measured when investigating any new dataset. Other similarity metrics should also be considered. IBCF offers a small 0.05 RMSE advantage for the dataset explored, a trend held throughout all densities. Finally, isolating ideal k values for nearest neighbors was especially important. The ideal parameter setting was remarkably consistent across all metrics.

One possible future bit of work would be to explore interpolation of missing values in the user-item or item-item matrix. In this dataset, the item-item matrix gave better results and consumes far less resource. It might only be feasible when exposed to the full dataset.

Optimizing algorithms proved important for runtimes. Cache aware data structures reduced run times by over 60% from less aware approaches, moving run times from 3 to 5 days down to less than 1 day. Care also must be taken in using built-in data structures offered by a lanaguage. For example, while STL hashing might be useful in sparse density datasets, a flat matrix in memory for item-item proved a better choice with linear lookup times. This difference was decisive in choosing between the R *recommenderlab* approach or writing custom software. Another area of development would be to move data structures to GPU memory for GPU processing. At its core, the algorithms perform a large series of multiply-accumulate instructions which are exactly suited to GPU processing. The speed-up could be remarkable, enabling a far more expansive view of data mining algorithms.

GIT Repository

Code for this project can be found under the 'c++' subdirectory at the Github address `https://github.com/AquaPi271/alda_project`.

References

[1] Xavier Amatriain, Justin Basilico. *Netflix Recommendations: Beyond the 5 Stars (Two Parts)*. The Netflix Tech Blog, 2012. https://medium.com/netflix-techblog/netflix-recommendations-beyond-the-5-stars-part-1-55838468f429. 11/5/2017.

[2] Linyuan Lu, Matus Medo, Chi Ho Yeung, Yi-Cheng Zhang, Zi-Ke Zhang, Tao Zhou. (2012) *Recommender systems*. In Physics Reports pp. 1-49, Elsevier B. V. 0370-1573.

[3] Yehuda Koren. (2009). The BellKor Solution to the Netflix Grand Prize.

[4] Robert M. Bell, Yehuda Koren. (2007) *Scalable Collaborative Filtering with Jointly Derived Neighborhood Interpolation Weights*. Seventh IEEE International Conference on Data Mining. pp. 43-52. IEEE.

[5] Michael Hahsler (2017). recommenderlab: Lab for Developing and Testing Recommender Algorithms. R package version 0.2-2. http://lyle.smu.edu/IDA/recommenderlab/

[6] Daniel Lemire, Anna Maclachlan. (2007). *Slope One Predictors for Online Rating-Based Collaborative Filtering*. In CoRR, Volume abs/cs/0702144.

[7] Joonseok Lee, Mingxuan Sun, Guy Lebanon. (2012). A Comparative Study of Collaborative Filtering Algorithms.