Global Blending on Sparse Ratings in the Netflix Prize 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 employs are 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 (need citation). 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.

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 recommends based on movie similarity. 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[1].

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. The scope is beyond what can be provided in less than a semester's time [2].

It was noted in "BellKor's Pragmatic Chaos"'s work that data sparsity was the main obstacle to address[2]. This project will focus on exploring remedies to this issue by using bulk information from the dataset when insufficient detailed information is in short supply. When adequate nearest neighbor information is available, the similarity rating will be used. If the nearest neighbor threshold falls below some value, overall average and variance for movie will be weighted and added to produce the targeted rating. Timestamps of ratings may also be considered to see if they can be blended into weights to improve target ratings.

The setup will rely on the *recommenderlab* tool suite[4]. The purpose of this suite to to expose a sandbox experimental environment for recommender systems. Support is provided for UBCF and IBCF systems as well as variants of singular value decomposition (SVD) and hybrid solutions.

The dataset also provides timestamp of rating, title of movie, and DVD release date of movie. Briefly, a bag-of-words approach was considered for identifying movie genres but was rejected outright because of too many proper names. Additional support will be provided by C++ routines used to clean data, and provide training and test sets.

2 Method

The dataset is comprised of over 100 million ratings of 17700 movies from 480,000 users. 17700 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.

The overall experiment will follow the guidelines for evaluation as set by Netflix. A test set (not provided) will query the recommender system which will give a decimal rating for the user based on that user's prior movie ratings. Netflix secretly held the actual rating given by the user. Errors will be tabulated using the root-mean-square-error formula:

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

While the data are considered clean by Netflix, it is enormous, occupying over 1.5 GB of space on 17700 separate text files. Exploration of the space revealed that for many of the functions in *recommenderlab*, the memory constraints were insufficient to hold the data. To get around this issue, a computer program was developed to sample recommendations from the overall merged dataset. Parameters given include the number of users, number of movies, and random seed. The program runs in two passes. The first pass collects all of the users from the randomly chosen movies. From this user list, random users are chosen up to the input parameter. A file is then produced with these sampled selections. The file is passed to another program which measures bulk statistics to ensure ratings follow a similar distribution as the overall dataset. Additional user metrics are compared to ensure the number of user ratings scales with the diminished movie count.

From the sampled data set, RMSE will be computed with k=10 cross-validations (k value subject to change pending on runtime) against each global recommender method. This procedure will be replicated across all valid sample set and serves as a replacement of Netflix's hidden testset.

A first priority is to establish rankings based on bulk metrics through average ratings. This is the starting point for the Netflix prize contributions. This technique yields 1.05475 RMSE, similar to other contestant's findings.

From here IBCF and UBCF models will be explored. Parameters considered will be number of nearest neighbors (k-NN value), similarity metric (cosine vs. Pearson), normalization, recency of

ratings, and blending of global movie parameters for sparse queries[3]. Time permitting, the Slope-one method of estimation will be explored. If possible, a hybrid model will be constructed that blends various combinations of the above into an overall model. Each technique is briefly discussed.

2.1 UBCF

This technique is given a targeted user that requests a rating on a particular item. Based on this user's prior ratings, a search is conducted across the entire user space to find k nearest neighbors who are most similar to the given user. These users must also score a rating for the targeted user's requests. The targeted ratings are then averaged to estimate the rating. The user space is large potentially making this search expensive.

2.2 IBCF

Similar to UBCF, IBCF instead seeks items that are most similar to each other. The first step is different. All items are compared against each other such that two separate users have a rating for both items. A similarity metric is scored across all such matching measurements. For n items this will produce $n(n-1)/2$ unique entries. The user's score will match up to k nearest neighbor items where a final rating will be averaged across these items. Because there is a relatively small number of items in comparison to the user space, the representation can remain inside memory for blended approaches.

2.3 Similarity Metrics: Cosine and Pearson

Similarity metrics 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.

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.

2.5 Recency of Ratings

Humans, being what they are, have limited memories. A bias might be revealed that involves the time between ratings. For UBCF and IBCF a weighing factor will be used amongst the k nearest neighbors. Another approach might be to explore if a user's bias changes over time.

2.6 Blending of Global Data for Cold Starts or Sparse Matches

In sparse data sets, enough missing data could produce less than 'k' nearest neighbors. Or the quality of the nearest neighbors may be lacking. Below a certain threshold of quality matches, the estimate rating will be blended with the global rating for that movie. The amount of weight given to the global rating will be inversely proportional to the number of quality neighbors below the k threshold. The

same rationale can be applied to the *Cold Start* problem. That is, a new user with few ratings offers only a small matching sample space.

2.7 Slope-one Estimation

Slope-one Estimation is another area to explore[5]. This technique simplifies UBCF and IBCF by forcing the linear approximators to use a slope of 1 against one free parameter. This free parameter is computed by taking the average difference of ratings between two items or two users. It is then added to the known rating to produce the targeted result. Slope-one is advertised as a technique to reduce overfitting.

3 Experiment

Under Construction.

4 Conclusion

The dataset is large. A good portion of time was spent on exploring IBCF and UBCF features in *recommenderlab* by using the built-in MovieLense database. The routines did a great job in explaining the overall programming approach, but they fail to run against extraordinarily large amounts of data. The remedy is to sample the full dataset with a collection of C++ tools. To this end, much of the development work has gone into creating these tools and processing the data for use in R Studio.

The toolsuite might be extended to perform the calculations described in this report. Raw C++ can handle the large amounts of data and run much quicker than R's interpreted environment. However, the interpreted environment offers the advantage of operating on data without reloading it for each experiment.

A day or two might be spent looking for APIs to enhance the approach.

Under Construction.

References / Papers to Read

[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] 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.

[3] 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.

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

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