Time Influence on Ratings in the Netflix Prize Dataset

Timothy Ozimek
North Carolina State University
teozimek@ncsu.edu

1 Background

The online movie service, Netflix, gives a user the choice of watching amongst several thousand movie titles. While not all titles are available at all times, Netflix will rotate selections, and add new ones, during three month windows. Netflix offers a recommendation system that suggests movies that suits a user's tastes. This feature works especially well after a user has recorded a rating after they watch a movie. The more ratings they score, the better the suggestion becomes.

One way of understanding the recommendation system is through everyday word-of-mouth conversation. Friends and acquaintances tend to share common opinions over a variety of topics. Likes and dislikes accordingly tend to coincide. The field of data mining and machine learning has observed this relationship and formalized the task as a recommender system (need citation). Such systems tend to employ the technique of collaborative filtering. In short, this method scours data, looking for commonalities in a request. Items deemed "close enough" to a request are filtered and used to predict. This filtered group may contain ratings not in the initial request. Based on these ratings an estimate is constructed for how a given request might rate that item. In this way, recommendation systems offer new knowledge.

The Netflix Prize dataset was supplied by the Netflix movie corporation in 2006 for use in a prediction contest. The objective of the contest was to predict how a user would rate a movie and compare through the root-mean-square-error (RMSE) metric at how well the scheme worked. The data set is vast, occupying over 100 million ratings from 480,000 users on 17700 movies. The objective was to beat Netflix's proprietary system, *Cinematch*, by 10 % on the RMSE. metric. Part way through the contest a Progress Prize was awarded because of an 8.43% improvement. This result was produced after 2000 hours of work with the use of a combination of 107 algorithms[1].

Netflix's services have since changed from mail-order DVDs to a streaming service. As a result, the fundamental recommendation system required new adaptations; the final prize's full algorithm was not used because it was applicable to the old service. However, two main algorithms, singular-value decomposition (SVD) and restricted Boltzmann machines (RBM) found useful roles under Netflix's expanding recommendation system[1].

The objective of this paper is less ambitious than the Netflix Prize. It will explore user-based collaborative filtering (UBCF), item-based collaborative filtering (IBCF) as described in [2], and other techniques through the *recommenderlab*[3] R framework and custom C++ software. It will attempt to implement some variations described in the Netflix Prize winners' preliminary paper found in [4].

2 Method

The dataset is comprised of over 100 million ratings of 17700 movies from 480,000 users. 17700 text files, one per movie, contains 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 rated

in the given data set. 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.

In other data sets, such as MovieLense provided as a sample in recommenderlab[3], additional information is often provided in the form of genre, director, reviews, et. al. No such telling features exist with the Netflix data. While harvesting this data from the web is certainly possible, it was not an approach taken during the contest, which intended the dataset to be self-contained. Furthermore, the movie names limited options through bag-of-words approaches.

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:

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

For cold starts a new approach will be taken. Up to a certain number X of prior ratings the global average will be blended with similarity. The weight will decrease by some function f(x), perhaps linearly, and will use variation on the given parameter to attenuate f(x).

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 of the file 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 sets.

3 Experiment

4 Conclusion

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

[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.