Integrating Contextual Information in Recommendation Systems

Ammar Raşid
Özyeğin University
ammar.rasid@ozu.edu.tr

Abstract

Recommendation systems are one of the most used applications of data science. In this paper, we show that considering a user's reliability in rating a movie contributes to the quality of recommendations. Furthermore, we introduce two user-based similarity metrics that show significantly better performance than our baseline, paerson correlation.
Keywords: Recommendation Systems, Collaborative Filtering, Similarity metrics

1 Introduction

People have long been consulting their social circle for recommendations. With the advent of online shopping and marketing, recommendation engines became a necessity. Recommendations are driven by the assumption that consumers behaviour follow detectable patterns, collectively and individually. Some recomendation systems recommend most popular items among all users. More "relevant" recommendations are those most popular items among the group of users similar to the user looking for recommendations.
Recommendation problems are generally modelled as a set of customers, a set of items, and a set of ratings. A utility many-to-many function is used to map customers to items, with or without ratings. Given a set of known ratings, the goal is to infer the unkown ratings.
Content-based recommendation engines put emphasis on the history of a customer's ratings. Items similar to those highly rated by a user, are likely to harness similar rating from the user, and are thus considered a potential recommendation. Another approach is based on association rules. Items consumed together more frequently, tend to be similar. If a user liked (or disliked) a certain group of items, s/he is likely to also like (or dislike) similar items. In general, collaborative filtering techniques recommend items similar to other items previously liked, by the query user, or by users similar to the query user.

2 Data and denotations

The data we are using is MovieLens-20M [2]. It is composed of $20,000,263$ ratings by $138,493$ users on $26,744$ movies of various genres. The dataset includes genome scores of tags users have assigned to different movies and the corresponding tag-movie relevance score. Only $10,381$ movies have been tagged. Let R be the set of ratings, U be the set of users, m be the number of users, I be the set of items, and n be the number of items. I_u denotes the set of items rated by user u. $R_{u,i}$ denotes user u's rating of item i. We will denote the rating high scale as β, which in *MovieLens* equals 5.0.

3 Ratings refinement

We use the tag-movie relevance score as our anchor for calculating users' reliability. We define reliability of a user in equation 1

$$\Omega_u = \frac{1}{L} \sum_{\forall r \in R_u \subset R} \sigma(F(r,\tau)) \tag{1}$$

Where r is the relevance score of each tag assignment by user u. R_u is the set of relevance scores of tag assignments by user u, which is a subset of R, the set of all relevenance scores. L is the length of the subset R_u. Tag assignments is transformed so that relevance scores less than or equal to τ will be practically nullified. The detailed steps of the linear transformer F is layed in 2.

$$\begin{aligned} F(r,\tau) &= a \times r - b \\ a \times 1 - b &= 6 \\ a \times \tau - b &= -6 \\ a &= \frac{12}{1-\tau} \\ F(r,\tau) &= \frac{12 \times (r-1)}{1-\tau} + 6 \end{aligned} \tag{2}$$

The sigmoid function is approximately 1 at $x = 6$, and 0 at $x = -6$. Therefore, relevance scores equal to or less than τ will be transformed to -6 or less respectively, and in turn be squashed to 0 by the sigmoid function. We chose $\tau = 0.2$ for purely empirical reasons.

4 Alternatig Least Square

Alternating least square (ALS) is one of the most common collaborative filtering based recommendatıon systems. ALS uses both matrix factorization and Oridnary Least Square (OLS) optimization. The ratings matrix R, defined in equation 3, can be factorized as $U \in R^{m \times k}$ and $P \in R^{n \times k}$, where $R \approx U \cdot P^{\intercal}$, and k is the rank under the assumption that R is affected by k effects.

$$R_{ui} = \begin{cases} r, & \text{if user } u \text{ has rated item } i \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

The cost function for matrix factorization is defined in equation 4

$$J = ||R - U \times P^{\intercal}||_2 + \lambda(||U||_2 + ||P||_2) \tag{4}$$

Ordinary least square uses pseudo inverse techniqueto solve a linear system, e.g. $Ax = b$, as shown in equation 5.

$$x = (A^{\intercal}A)^{-1}A^{\intercal}b \tag{5}$$

Alternating least square is an iterative optimization process of two-steps. One step is to fix P and solve for U, and the other is just the opposite. Assuming the vector representing the ratings of a specific user u_i is independant of similar vectors for other users or $u_j \neq i$, and similarly for items, this makes ALS highly parallelizable.

5 Collaborative filtering approaches

Collaborative filtering approaches have been commonly used for recommendations, either solely or together with other techniques. The core of any collaborative filtering algorithm is the similarity metric used to create a users (or items) similarity space. Since Jaccard similarity ignores the value of ratings, and cosine similarity treats missing ratings as negative, we chose pearson correlation coefficient as our baseline similarity metric. A query user u_q rating of an item i is the average of all other users' $u_{p \neq q}$ ratings of the same movie i weighted by $\Psi(u_q, u_p)$, where Ψ is the similarity function. See equation 6.

$$\hat{R}_{u_q,i} = \frac{1}{N} \sum_{\forall p \neq q \in U} R_{u_p,i} \times \Psi(u_q, u_p) \tag{6}$$

N is the number of ratings of item i by all users other than the query user u_q.

5.1 Bipartite Graph Reinforcement

We model the user-item interactions as a bidirectional bipartite graph, where users *map* to items, and vice versa. Similar approach has been used by [1] for computing a user similarity space for stance detection purposes. This allows us to use the bipartite graph in conjunction with graph reinforcement to estimate the conditional probability that a user u_q would map to user u_p, i.e. $p(u_p|u_q)$. Given a user u_q we traverse to an item i to compute $p(i|u_q)$, i.e. the likelihood of user u_q's rating of item i given all the rating points user u_q has given, equation 7.

$$p(i|u_q) = \frac{R_{u_q,i}}{\sum_{\forall j \in items} R_{u_q,j}} \tag{7}$$

We then traverse from item i to user $u_{p \neq q}$ to compute $p(u_p|i)$, i.e. the likelihood of user u_p's rating of item i given all the rating points item i has collected, equation 8.

$$p(u_p|i) = \frac{R_{u_p,i}}{\sum_{\forall u \in users} R_{u_p,i}} \tag{8}$$

If there is only one item i rated by both u_q and u_p, then $p(u_p|u_q) = p(i|u_q) \times p(u_p|i)$. However, to account for all mutually rated movies, we compute the conditional probability $p(u_q|u_p)$ as shown in equation 9. Finally, $p(u_p|u_q)$ is normalized by the number of the items rated by both u_q and u_p. The **BGR** similarity function is formally defined in equation 10

$$p(u_p|u_q) = 1 - \prod_{\forall u_q, u_p \in U, \forall i \in I} (1 - p(i|u_q)p(u_p|i)) \tag{9}$$

$$\mathbf{BGR}(u_q, u_p) = p(u_q|u_p)^{\frac{1}{M_{u_q,u_p}}} \tag{10}$$

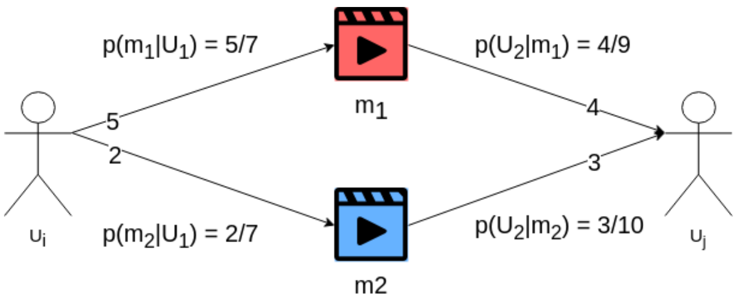

Figure 1: A simplified example illustrating BGR in action

5.2 Weighted Jaccard

Inspired by jaccard similarity, we introduce a similarity function that accounts for the inability of jaccard in capturing rating values, equation 11.

$$\mathbf{WJ}(u_p, u_q) = 1 - \sum_{\forall i \in I_q \cap I_p} \frac{abs(R_{q,i} - R_{p,i})}{\beta} \tag{11}$$

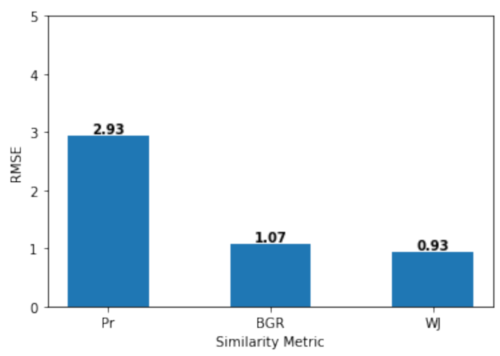

Figure 2: RMSE scores (lower better) of collaborative filtering recommendations with three interuser similarity metrics

6 Experiment

6.1 Framework

We opted to use Apache Spark distributed computing framework. We parsed the data as dataframes as a preprocessing step for dataframe operations. For calculating users similarity, we modeled the problem as a MapReduce pipeline, which we applied on the Reselient Distributed Dataset (RDD) format of the data. The system was distributed over 8 virtual cores, corresponding to 4 actual cores, each has base clock speed of 3.4 GHz.

6.2 ALS with ratings refinement

We use users' reliability scores we mentioned in section 3, to scale the ratings. Each rating is multiplied by the relibability score of the user who gave that rating. That introduced a substantial improvement on both train and test set. We trained ALS on 90% of the data, and tested on the remaining 10%. Using cross validation with parameters grid, we found that the best model has rank 14 and regularization parameter 0.06. Table 1 shows the Root Mean Square Error (RMSE) scores of (ALS) model with and without ratings refinement.

	Train	Test
Original	0.68	0.75
Reliable	**0.56**	**0.62**

Table 1: RMSE scores (lower better) of ALS model on train and test with and without ratings refinement. "Original" refers to unscaled ratings. "Reliable" refers to ratings scaled by reliability scores

6.3 Collaborative filtering approaches

In ths section, we compare the performance of our two proposed user-based similarity metric against pearson correlation coefficient as a well-established baseline. We choose a subset of ratings by using only the ratings by 10 users. Figure 2 shows the performance of the three similairty metrics on test set. Since all similarities are computed in nearly the same number of steps, the runtime comparison is negligible.

7 Conclusion and future work

We have shown that scaling the ratings by their critics' reliability scores substantially improves the performance of our ALS model. We can conclude that contextual information contributes to the improvement of recommendation systems. We have proposed two similarity metrics that have shown similar promising performance. We aim to study more metrics features to further polish our proposed similarity metrics and integrate more contextual information.

References

[1] Kareem Darwish, Walid Magdy, and Tahar Zanouda. Improved stance prediction in a user similarity feature space. pages 145–148, 07 2017.

[2] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4):19:1–19:19, December 2015.