Social and Drift-Aware Methods for Collaborative Filtering

Samuel Taylor

2 May 2016

Contents

1 Abstract

Collaborative filtering (CF) is a popular approach for recommender systems. This approach assumes that users often like items that similar users like. Similarity between users is usually a function of the users' ratings of individual items. With the assumption that users who are closer together on the social graph are more similar, we develop a CF algorithm that takes social data into account via a new similarity function.

Content-based filtering is another approach to recommender systems. This approach explicitly models users' preferences about items. Some work research has been done to model the way users' preferences shift over time. We attempt to apply the idea of shifting preferences to the way CF finds similar users.

Through comparison of ROC curves, we find that using a similarity function based on social data yields a better predictor than one which uses a similarity function based on users' ratings in some circumstances. Our drift-aware method was worse than a baseline implementation in all circumstances.

2 Introduction

Recommender systems have many applications. Among such applications, they are used to recommend music to users as well as decide which advertisements to display to which users. Two major types of algorithms for recommender systems are content-based filtering and collaborative filtering. The difference between these approaches is well described by comparing the music recommendation algorithms used by Last.fm and Pandora.

Last.fm is an example of collaborative filtering, and Pandora is an example of content-based filtering. Last.fm produces recommendations for a given user by finding users who are similar to that user and then finding other artists those users commonly listen to. By contrast, Pandora has descriptions of the musical elements of each artist (e.g. Artist A has "pop elements, catchy riffs, and female vocals"). Pandora uses a user's listening history to determine what musical elements are most predictive that a user will like an artist. Then, it finds and recommends artists that have those musical elements. For this paper, we focus on collaborative filtering (often abbreviated "CF").

This problem has an input space consisting of (user, item) tuples and an output space consisting of the ratings given by users for the items. To be more specific, we can say that the input space $X = \{(u_i, p_j) | u_i \text{ is a user and } p_j \text{ is an item}\}$. Further, the output space $y = \{r_{i,j} | \text{ user } i \text{ gave item } j \text{ rating } r_{ij}\}$.

A CF system calculates a weighted average over all users of the user's deviation from their mean rating. For this weighted average, each weight is the similarity between the input user and the other user. More formally, our predicted rating for the active user a on item i, $p_{a,i}$, is calculated:

$$p_{a,i} = \overline{r}_a + \frac{\sum_{u=1}^{n} (r_{u,i} - \overline{r}_u)\, w_{a,u}}{\sum_{u=1}^{n} w_{a,u}}$$

For this formula, $\overline{r}_u$ is the mean rating given by user u. $r_{u,i}$ is the rating given by user u to item i. Finally, $w_{a,u}$ is the weight (or similarity) between the active user (that is, the

user for whom we are predicting a rating) and user u.

3 Related Work

Herlocker et al. outline a framework for collaborative filtering [1]. In it, they discuss in-memory collaborative filtering, wherein a database of users and their ratings is kept in memory. The algorithm then uses this database to compute recommendations with a similarity-based method (as described by the formula for $p_{a,i}$ above).

Massa and Avesani look into trust-aware recommender systems[3], which take into account explicit statements about the trust users have in other users' ratings. For instance, given a system where users can trust other users on the interval $[0, 1]$, Alice can express, "I trust Bob with certainty 0.8". This statement would be a way for Alice to tell the recommender system that Bob's taste and her taste are well-aligned. While these "trust networks" are similar to the social networks we are researching, they are different. Trust networks are explicitly related to users' ratings, while social networks are not necessarily related to users' preferences.

Koychev and Schwab research drift in a user's interest in a content-based filtering system [2]. By weighting more recent ratings more heavily in their algorithm, they are able to model the fact that users' interests change over time.

4 Methodology

4.1 Algorithms

Three approaches have been implemented.

First, a baseline implementation of collaborative filtering was made using the similarity function $s(u_i, u_j) = \exp(m/\epsilon)$, where m is the number of artists that have been tagged by both users and ϵ is a normalization constant that is learned from the training data. The logic behind this similarity function is that if two users have both tagged many artists, they are much more similar to each other than two users who have both tagged relatively few artists.

Second, an implementation of collaborative filtering which takes social information into account was made. A graph was constructed where each user is a node and each friendship is an edge between the users who are friends. After using the Floyd-Warshall algorithm to find the shortest paths between all pairs of users, we can define similarity. If $d_{i,j}$ is the length of the shortest path between user i and user j, the similarity function in this algorithm is $s(u_i, u_j) = \exp(-d_{i,j}^2/\epsilon)$. Again, ϵ is a normalization constant learned from the training data. The logic behind this similarity function is that if users are very close on the social graph (e.g. one or two hops away), then they are much more similar than users who are very far apart on the social graph (e.g. six or seven hops away).

Finally, an implementation of collaborative filtering which takes into account drift in user interest was made. For this approach, we again implement a new similarity function. The similarity function we use is closely related to the Euclidean distance between the timestamps of the tags of the artists both users have tagged. More formally, let A be the set of artists

tagged by both user i and user j. Then, let $r_{i,A}$ be a vector wherein each element is the rating given by user i to an artist in A. $r_{j,A}$ is a similar vector, but with the ratings of user j. Assuming the ratings of each artist are ordered in the same way in each vector, we can denote the Euclidean distance between them as $|r_{j,A} - r_{i,A}|^2$. If we let $d(i,j) = |r_{j,A} - r_{i,A}|^2$, then we can define the similarity between user i and user j as $\max_{k,m}(d(k,m)) - d(i,j)$. That is, the similarity is the maximum distance between any two users minus the distance between user i and user j.

The idea behind this similarity function is that users who tag artists at similar times will have a larger similarity, and users who tagged artists at wildly different times will have a smaller similarity.

5 Experiments

5.1 Dataset

To evaluate the comparative performance of each algorithm, we used a dataset from the HetRec 2011 conference drawn from Last.fm. This dataset includes social data in the form of user-user pairs, where a pair signifies that users are "friends."

On the Last.fm dataset, the "items" to which the users assign ratings are artists. While the dataset does include record of how many times users have listened to each artist (which could be thought of as a rating), these data points don't have an associated timestamp. We could implement a baseline collaborative filtering algorithm as well as a social-aware version on this data, but we would not be able to use a drift-aware approach on this data. As such, in order to compare across each of the three (eventual) methods using the same dataset, it's best that we use data with timestamps.

The Last.fm tag-assignment data does have timestamps. This dataset consists of (user, artist, tagID, timestamp) tuples, representing that user tagged artist with tagID at timestamp. Considering that a user who takes the time to tag an artist probably likes them, we equate a user having tagged an artist with the user assigning the artist a rating of +1. If a user has not tagged an artist, we consider them to have rated the artist -1.

5.2 Evaluation

From our dataset $\mathcal{D}$, we find the most popular TODO tags. Then, we limit the dataset of tag assignments to only those where a user tags an artist with one of those popular tags. We randomly select 80% of the data for training, and withold the other 20% for testing. After training each algorithm on the training data and finding its predictions on the test set, we look at the ROC curve for the algorithm.

5.3 Results

6 Analysis

The social data-based approach outperforms the baseline implementation at certain points on the ROC curve. Overall, the area under the ROC curve is higher for the approach taking into account social data than the area under the ROC curve for the baseline CF algorithm. This result is surprising, as a social network is not necessarily related to users' musical preferences.

At the same time (and to equivocate on the word "trust" a little bit), people tend to trust their friends. Perhaps, then, a social network represents an analog for a trust network. Were this the case, then systems may be able to gain some of the benefits of trust networks from pre-existing social data. Consider the case of an online service which doesn't collect data to build a trust network, but does have data about the friends users have added on the service. Instead of implementing a trust data-gathering feature, the owners of the service could potentially use social data to improve their service's recommendations at a lower cost (both in terms of resources spent implementing the feature and in terms of getting their users to answer questions about trust).

7 Conclusion

We have presented a potential improvement to collaborative filtering systems through usage of social data. By empirical analysis on a dataset from Last.fm, we show that this approach can be effective.

Our experiments with application of drifting user interests to CF are less clear. The data about the time at which users tagged artists may not be informative in the context of generating recommendations that are not time-sensitive. This area may warrant future study.

References

[1] Herlocker, J. L., Konstan, J. A., Borchers, A., and Riedl, J. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval* (1999), ACM, pp. 230–237.

[2] Koychev, I., and Schwab, I. Adaptation to drifting users interests. In *Proceedings of ECML2000 Workshop: Machine Learning in New Information Age* (2000), pp. 39–46.

[3] Massa, P., and Avesani, P. Trust-aware recommender systems. In *Proceedings of the 2007 ACM conference on Recommender systems* (2007), ACM, pp. 17–24.