An Online Social Network-based Recommendation System

Jorge Aranda, Inmar Givoni, Jeremy Handcock, Danny Tarlow
Department of Computer Science
University of Toronto
10 King's College Road, Toronto, Ontario, Canada, M5S 3G4

Abstract

We present a social network-based recommendation system that uses data from user profiles and user-to-user connections. We adapted our implementation to use data from the BoardGameGeek (BGG) website. The end result system will be usable by members of BGG to get recommendations given both their ratings history and their friend relationships.

At this stage, we have gathered all of the needed data from the BGG website, and we have started to run the algorithm for our recommendation service. The remaining steps are to run experiments to determine suitable parameter settings for the algorithm, and create a web-based application.

1 INTRODUCTION

With the rise of the web-based economy in the past 10 years, internet retailers have begun to find it practical to use algorithmic approaches to decide on content to show users. If a retailer can show a user more relevant content, then the user will be more likely to buy or otherwise use the service. Perhaps the most widespread and successful class of these algorithmic solutions are recommendation systems.

Recommendation systems on the web were first popularized by Amazon.com, which would show users personalized recommendations of items that the system thought they would like based on the items that they had bought or rated in the past (Linden, et. al. 2003). Since then, the practice has spread widely, as computing power becomes cheaper and as the algorithms become more widespread.

As research progresses on building faster and better recommendation systems, a fundamental limitation begins to appear, which is that there is only so much information contained in a user's rating history. Regardless of the algorithm, it will be impossible to pass certain baselines without incorporating additional information about the items and the users involved in the system. One direction that has received attention in the last couple years is to do "content-based" collaborative filtering, which treats items as objects with many attributes, some of which are shared across items (Garden and Dudek, 2006). The goal of the system is to figure out which attributes about items a user likes, and to figure out which attributes an item has (usually by asking users to provide this information as well).

An approach that has received less attention is to use structure on the users as an additional source of information. This is the problem we want to address. One reason why this approach has received less attention is likely because the data is harder to come by. In traditional recommendation systems, data can be made public without too much worry about privacy, but as descriptions of users become more complex, the risk of privacy breaches rises. Thus, the companies that do have access to both user histories and relationships between users are unlikely to share their data.

The problem is difficult because good data sets are hard to come by and because traditional algorithms assume that users are independent. In order to make use of the additional information that is available, we must develop a new algorithm that uses this information in a way that improves results, and in a way that that can be implemented efficiently. Our project addresses exactly these two problems.

We describe the sociological and machine learning foundations of the approach, and demonstrate it in the domain of recommendations for users of an OSN for boardgaming. Finally, we discuss the privacy issues that arise from the use of our algorithm.

2 BACKGROUND

Recommendations in social networks

There are findings in the sociological and psychological disciplines that point to the relevance of a person's social network in determining their tastes, preferences, and activities. The principle of *homophily*, for instance, is well established in the Social Networks field. McPherson *et al.* reported how "similarity breeds connection". They discovered that "people's personal networks are homogeneous with regard to many sociodemographic, behavioral, and intrapersonal characteristics". In other words, we share many attributes with the people close to us. Reversing this principle suggests that, if we have information about the connections in a person's network, we can infer some of the person's attributes.

It is possible that at least some of the similarities within a network are caused by the influence and interactions of the people in the network. People tend to remember information that was concretely given to them (that is, in personal interactions) better than abstract information (like statistical base rates). For example, Hogarth states that when considering to buy a certain car model we will likely give more thought to the direct advice of a friend than to each of the 100 respondents to a survey in a specialized magazine.

More specifically, Leskovec *et al.* discuss the phenomenon of information cascades, in which individuals adopt a new action or idea due to influence by others. In the most extreme cases, knowledge about a full network's behaviour determines the behaviour of its members –making a "top hits" list available in a music downloading website affects the popularity of the songs, and several different networks, kept in isolation of each other, prefer completely different songs, to the point that it is impossible to predict which will be the most popular songs for a network without observing the behaviour of the users in the network.

Our domain: BoardGameGeek

BoardGameGeek (BGG) is the most popular community for people that play board and cardgames. It hosts a database of more than thirty thousand games, along with their corresponding reviews, photos, rules clarifications, ratings, player aids, and other user-generated content. Over forty thousand registered users have rated games in this website, and many of them have recorded some of their personal information in their profiles and made explicit their friendship links ('GeekBuddies', in BGG parlance) with other users. The relative openness of the BGG database (the website provides an XML API that gives access to some of its data), as well as its moderate size made it an appropriate choice to try our social-based recommendations algorithm.

3 DATA COLLECTION

We intended to obtain our data directly from the BGG system administrators. We e-mailed them twice, but received no response from them. Therefore, we decided to write a set of crawlers to gather the data.

At this point, we have finished collecting the data from the BGG website. We crawled for the following data:

- Games that any user has rated (about 30K)
- Users that have rated any games (about 40K)
- Ratings given by all users to any games
- Games in all users' wishlists, as well as the weight given to wishlist items by each user
- Links between users ("GeekBuddies")

To get the user information and ratings we used an API provided by the BGG website. However, getting game and GeekBuddy data was more problematic, as it was only available to logged-in users. To overcome this our crawler had to "trick" the website, passing as a logged-in user. We sent requests to the BGG website every 2 seconds; a complete pass to crawl the data we need takes slightly more than 3 days if run sequentially.

We saved the collected data in a database, and wrote scripts to generate flat text files with sparse representations of information to be used by the machine learning algorithm.

4 ALGORITHM

We have chosen to implement a modified version of probabilistic matrix factorization (PMF [Minh 07]). Different variants of this approach are very popular among Netflix competitors [Netflix.com]. The algorithm takes as input the $M \times N$ rating matrix R, where R_{ij} is the rating user i gave to game j. The basic idea behind PMF is to find a low-rank decomposition of R by approximating it as a product of two low-rank matrices:

$$R \approx UG \tag{1}$$

Where $U \in \mathcal{R}^{M \times D}$, $G \in \mathcal{R}^{D \times N}$, and D is intentionally kept small ($D \ll M, N$). An intuitive explanation for this kind of approach is as follows. We imagine there exist only a small number of prototypical users in the world. Each one is associated with her particular profile of game ratings. An example of such prototypical user is an ardent war-games lover, another might be a person who most enjoys word-based games. These profiles are stacked one on top of the other to create our G matrix. It is clear then , that D represents the number of prototypical users we believe to exist, and is a model parameter that needs to be tuned. It is important it is kept small to avoid data overfitting[1]. Now, each real user is explained as some weighted combination of this small set of profiles. By mixing the profiles we can 'span' the space of all real users. Each user is associated with D numbers that tell us how to mix the prototypical profiles together in order to generate that user's ratings profile. These length D vectors stacked on top of each other give us the U matrix. The task of the algorithm is to determines the set of prototypical profiles and the linear combination coefficients associated with each user, or in other words, find U and G that when multiplied together come as close as possible to the original matrix R. There are many matrix factorization approaches that seek a low-rank approximation (i.e. FA and PCA [Jolliffe, 86]). PMF is particulary suited for very sparse input matrices since it minimizes the squared error $(\sum_{i,j} R_{ij} - U_i^T G_j)^2$ only over the non-zero R entries. In such a model, once we determine U and G, we can easily find out what ratings the model predicts users will give to games they have not ranked (the empty entries of the original R). Then we can choose the games with the highest predicted ratings and recommend them to the users.

In order to incorporate the social network information into the model we introduce the $M \times M$ friendship matrix F. $F_{i,j} = 1$ if user i listed user j as a geek buddy[2] and zero otherwise. We then formulate the following model of the data:

$$R \approx FUG \tag{2}$$

In other words, each user 'internal' ratings profile is originally that linear combination of the prototypical users' profiles. However, he is 'externally' affected by his friends' profiles as well and they are added onto his. The friendship matrix simply adds that external influence. Since the F is known, we can easily find U and G by solving the problem

$$F^{-1}R \approx UG \tag{3}$$

[1]Overfitting is a term that refers to learning a model that can predict its input extremely well but does poorly on any new data. For PMF, the extent of overfitting is determined by the control knob over D

[2]Note that in BGG friendships are directed relationship, and are not necessarily symmetric

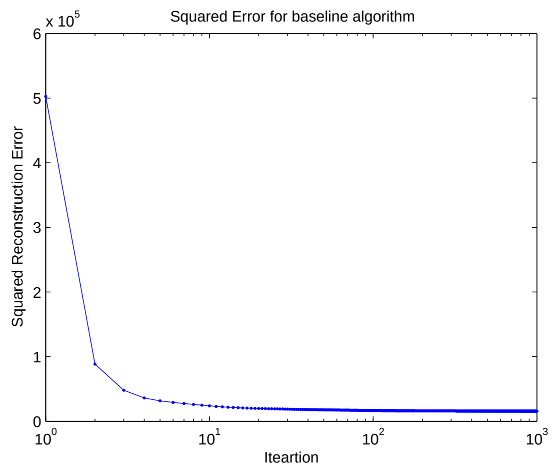

Figure 1: Decreasing sum of squared error during learning process

Recommendations can be obtained by calculating $(\hat{R}) = FUG$ and proceeding, as before, to reccomend games users have not rated which are predicted to have high ratings.

We have implemented PMF in java based on a matlab implementation provided to us by the authors. Additional algorithmic and implementation details can be found in appendix 7.

An interesting question in the context of social network information is whether we can implement the algorithm in a distributed manner so that users do not have to enclose friendship information. We assume for the sake of argument a setting where one's geek buddies are private information. We discuss the implications of such a requirement in appendix 7.

5 EXPERIMENTAL EVALUATION

The base-line version of the algorithm is complete (not using friendship information) and we are able to show it consistently decreases the average squared error.

We will use some of the collected information as a held-out test set and find out whether we can achieve any improvement in predicting user ratings compared to the base-line algorithm. We would like to test different settings of how we construct the entries of the friendship matrix. For example, it might be more sensible to have 1 on the diagonal entries of F, and some value between 0 and 1 for friendship information, indicating that friends are not as equally influential as one's 'internal' preferences.

We have written matlab scripts that randomly generate splits of the data to use as training and testing data. The idea is to hold out parts of the data and evaluate performance on the held out part. We are currently resolving numerical precision issues related to calculation of the inverse of the sparse matrix F matrix, and are meanwhile writing the scripts that will run the comparisons.

6 APPLICATION

The end result of our project will be a web-based application in which users of the BGG website can log in and get recommendations from our algorithm.

The service will work as follows. We will have a job executing weekly to pull data from BGG, and passing it on to our recommendation algorithm. The algorithm will output weight matrices, which will be uploaded to our server. Whenever a user logs in to our website, the server will perform a matrix multiplication for that user, and present a set of recommendations for him. New users will be presented with a list of board games to rate, so that we can offer them some initial recommendations.

Ideally, linking this application with the BGG website would conveniently allow us to use dynamic data, and would be more convenient for BGG users. We will continue to attempt to communicate with the system administrators of the website to establish this link. If this is not possible, we will announce the application to the BGG users through the forums of the website.

7 CONCLUSIONS

This is a stub. In this final report, we will discuss our results and future directions for our application in this section.

References

R. Hogarth. *Judgment and Choice*, John Wiley and Sons, 1980.

J. Leskovec, A. Singh, and J. Kleinberg. Patterns of Influence in a Recommendation Network. Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), 2006.

M. McPherson, L. Smith-Lovin, and J.M. Cook. Birds of a Feather: Homophily in Social Networks. Annual Review of Sociology, 2001, 27:415-44.

A. Minh, *et al.* Probabilistic Matrix Factorization Applied to the Netflix Rating Prediction Problem. *To appear in NIPS 2007.*

A. Narayanan and V. Shmatikov. How to Break Anonymity of the Netflix Prize Dataset. *arXiv:cs/0610105v2 [cs.CR]*

Netflix forum. http://www.netflixprize.com/community/viewtopic.php?id=778

Board Game Geek. http://www.boardgamegeek.com/

IT Jolliffe. Principal Component Analysis. Springer-Verlag, New York NY, 1986.

R. Crammer *et al.*. Secure Distributed Linear Algebra in a Constant Number of Rounds. Lecture Notes in Computer Science, 2001.

M. Garden and G. Dudek. Mixed Collaborative and Content-Based Filtering with User-Contributed Semantic Features, AAAI, 2006.

G. Linden, B. Smith, J. York. Amazon.com Recommendations: Item-to-item Collaborative Filtering. IEEE Internet Computing Industry Report, 2003.

APPENDIX A: PMF

Stub - more information about the algorithm.

APPENDIX B: Secure Recommendations

Here we discuss the possibility of building a distributed version of our recommendation system that does not require users to disclose their friendship relationships to the server or to any other user. There are several operations that need to be described in a privacy conserving manner (ranging in

For a new user,with some ratings and no friends, securely getting recommendation based on the learned model is fairly straight-forward since his expected ratings do not involve friendship information, and thus the problem essentially reduces to distributed recommendation under the standard PMF model. The user obtains the G matrix, which does not reveal any social information[3], from the server. He then needs to solve a least square optimization problem

$$\min_x (\tilde{G}^T x - \tilde{r})^2 \tag{4}$$

where x is the desired recomendation vector for the user, $\tilde{G}$ is a submatrix of G composed only of these columns for which the user has provided ratings, and $\tilde{r}$ is a vector of the users's existing ratings. The user can easily compute the solution by calculating locally $x = (\tilde{G}\tilde{G}^T)^{-1}(\tilde{G})^T(\tilde{r})$

The next scenario is a new user with some ratings and some friends (who appeared in the original R matrix). Again, the user needs to solve a least squares problem, but this time he needs to know his friends information as well. In this case we also send the U matrix and he can retrieve the necessary rows of U without disclosing who his friends are and proceed.

The most difficult problem lies in the training phase of the algorithm. i.e. calculating

$$F^{-1} \approx UG \tag{5}$$

without knowing F, or its inverse. We assume each user maintains a secret vector of his friends, (there must be

[3]Although there are statistical de-anonimization methods which could perhaps shed some light on the F matrix based on U and G

some ordering on the users, that can be the users id which are not secret). It is a vector of length (num users) with ones at position i,i (user himself) and positions i,j for every friend of the user.

There are two issues that must be resolved. The first is calculating F^{-1} while conserving privacy. The second issue is that if the user simply sends us his respective $F^{-1}R$, we can find the solution for F^{-1}.

1. Calculating F^-1:

It appears there exist secure multi-party calculating (SMPC) protocols for linear algebraic operations. Such protocols usually treat the case where each participant has only one entry in the matrix. They allow calculating quantities such as the matrix determinant [Crammer 2001]. It might be possible to create a protocol for determining the matrix minors securely and calculating the inverse There also might be more straightforward ways of doing so. There remains to verify that knowing a whole vector, rather than a single entry does not reveal too much information.

2. Exposing $F^{-1}R$ to the server (so that over all the server has the full $F^{-1}R$) without compromising F.

In general if we know $aR = x$, and R. finding a is again a least square approximation. Assuming an underdetermined system there is an infinite number of solutions, but ones that are binary might be easy to check. The user might resort in this case, to removing some of the friendship entries when performing $F^{-1}R$ in order to increase the level of ambiguity so that there are many possible solutions with zero one entries. Although we lose some of the social leverage, we maintain privacy. The user the evaluate the ambiguity level (number of different solutions such that not all of them reveal a particular friend) for different removals of friends (since there are few friends, he can in theory evaluate all (n choose k) options and choose what to return.

In this setting we assume the server provider is the actual adversary, and to ensure the users the solution is valid we need to verify we cannot use the over all information to glean the friendship information the directionality of the friend's information come in handy in this case. If we knew symmetry hold that would have made the task easier. This is still work in progress.