Learning Collaborative Filtering and Its Application to People to People Recommendation in Social Networks

Xiongcai Cai, Michael Bain, Alfred Krzywicki, Wayne Wobcke, Yang Sok Kim, Paul Compton and Ashesh Mahidadia
School of Computer Science and Engineering, University of New South Wales, Sydney NSW 2052, Australia
{xcai,mike,alfredk,wobcke,yskim,compton,ashesh}@cse.unsw.edu.au

Abstract—Predicting people who other people may like has recently become an important task in many online social networks. Traditional collaborative filtering (CF) approaches are popular in recommender systems to effectively predict user preferences for items. One major problem in CF is computing similarity between users or items. Traditional CF methods often use heuristic methods to combine the ratings given to an item by similar users, which may not reflect the characteristics of the active user and can give unsatisfactory performance. In contrast to heuristic approaches we have developed CollabNet, a novel algorithm that uses gradient descent to learn the relative contributions of similar users or items to the ranking of recommendations produced by a recommender system, using weights to represent the contributions of similar users for each active user. We have applied CollabNet to the challenging problem of people to people recommendation in social networks, where people have a dual role as both “users” and “items”, e.g., both initiating and receiving communications, to recommend other users to a given user, based on user similarity in terms of both taste (whom they like) and attractiveness (who likes them). Evaluation of CollabNet recommendations on datasets from a commercial online social network shows improved performance over standard CF.

Keywords—Data Mining; Machine Learning; Recommender Systems; Collaborative Filtering

I. INTRODUCTION

Traditional recommender systems attempt to discover user preferences over items by modelling the relation between users and items. The aim is to recommend items that match the taste (likes or dislikes) of users in order to assist the active user, i.e., the user who will receive recommendations, to select items from an overwhelming set of choices. Such systems have many uses in e-commerce, subscription based services and other online applications, where provision of personalised suggestions is required. By applying recommendation techniques, it is possible to greatly increase the likelihood of the successful purchase of products or services, since services or products are personalised and presented to the active user using information obtained from the purchasing behaviour of like-minded users. In online applications with a very large number of choices where customer taste is important, personalised recommendation of items or people becomes essential.

Approaches to recommender systems can be categorised as content-based or collaborative filtering (CF) methods. In content-based methods [1]–[4], the user will be recommended items similar to those the user preferred in the past. Content-based methods analyse the descriptions of items and provided user ratings to infer a model for recommending additional items of interest. CF methods recommend items based on aggregated user preferences of those items, independent of the availability of item descriptions.

CF algorithms fall into two categories: memory-based approaches and model-based approaches. Memory-based approaches [5]–[8] use heuristics to make rating predictions based on the entire collection of items previously rated by users. The unknown rating value \( r_{c,s} \) of the active user \( c \) for an item \( s \) is typically computed as an aggregate of the ratings of users similar to \( c \) for the same item \( s \). This aggregate can be an average or, commonly, a weighted sum, where the weight is a distance that measures the similarity \( \text{sim}(c_1, c_2) \) between users \( c_1 \) and \( c_2 \). By using similarity as a weight, more similar users make a greater contribution to a predicted rating. Typically the similarity between two users is based on their ratings of items that both users have rated. Approaches to compute similarity include correlation [6], [8], cosine-based [5], [9], default voting, inverse user frequency, case amplification and weighted-majority prediction [5], [10]. Those approaches usually use heuristics to model the weights. However, a major problem with heuristically-based weighted sum approaches is that they do not take into account the fact that the similarity of users to one active user may be more reliable than their similarity to another. Taking this seriously means the contribution of similar users to the final rating used in recommendation should be adjusted for different active users. In this paper we address this problem by learning how to weight the contribution of similar users for each active user.

Model-based CF [5], [11]–[16] uses the collection of ratings to learn a model, which is then used to make rating predictions. Although model-based methods have reported higher accuracy of recommendation than memory-based approaches, there are also some limitations. Firstly, these methods are computationally expensive since they usually require all users and items to be used in creating models.
Secondly, they attempt to predict the rating of a user rather than correctly rank the items. This is not sufficient, however, since any method that correctly predicts all the ratings will correctly rank all the items, but two methods that predict the ratings equally well may perform differently in predicting the rankings [11]. Instead of using ratings, we directly learn a ranking using a pairwise learning to rank algorithm.

Memory-based approaches can be more computationally efficient compared to model-based approaches due to the reduced number of correlated users to be considered when making recommendations. However, they may not model well the underlying contribution since they are based on heuristics to compute weight values using pre-defined similarity measures. In this paper, we introduce CollabNet, a novel recommendation method combining the advantages of memory-based and model-based approaches with learning to rank techniques. The method learns the weight value of similarly-minded users directly from the training data to build a regression model to predict the preferences of the active user. More specifically, the method uses gradient descent methods to learn the underlying social collaboration behaviours relating to ranking recommendations. Thus it has the ability to adjust the model learned to reflect the characteristics of the data. Furthermore, model construction is based on optimising to correctly rank items rather than predict their rating, which makes the algorithm more effective. The main contributions of this paper include: (i) a novel method of learning CF by combining memory-based and model-based approaches based on learning to rank techniques; and (ii) the application of the proposed method to the recommendation of potential friends or partners in social networks.

The paper is organised as follows. Section II describes our CF framework. Section III develops a machine learning approach for learning CF. Section IV investigates our learning CF approach applied to people to people recommendation in social networks. Experimental evaluation is in Section V and conclusions are in Section VI.

II. STANDARD COLLABORATIVE FILTERING

Traditional CF can operate in two directions: user-based or item-based. User-based approaches look for users who share the same rating patterns with the active user (the user whom the prediction is for) and then use the ratings from like-minded users to calculate a prediction for the active user. Item-based methods such as that of Amazon.com [7] use an item-item matrix determining the similarity of items in terms of pairs of items purchased together, which is then used to infer the preference of the active user.

The most important step in both approaches is determining similarity. Two items are similar if both are selected together by a set of users. Alternatively, two users are similar if they both select the same set of items (i.e., they have similar taste). The underlying assumption of CF approaches is that those who agreed in the past tend to agree again in the future. User-based approaches assume that two users will like the same items if they are similar in taste, therefore will recommend to the active user an item selected by a similar user: $i \Rightarrow u : \exists s, (s \leftrightarrow s \rightarrow i)$, where $i \Rightarrow u$ denotes recommending $i$ to $u$, $s \leftrightarrow u$ denotes that $s$ is similar to $u$ and $s \rightarrow i$ represents that $s$ selected $i$. Item-based approaches assume items can be related by the fact that they are frequently selected together by users, and will recommend an item which is similar to items that the active user selected: $i \Rightarrow u : \exists s, (s \leftrightarrow i \land u \rightarrow s)$.

III. LEARNING COLLABORATIVE FILTERING

A. Modelling similar user contributions

In the following sections, we discuss CF approaches in the context of people to people recommendation, unless specified otherwise. This means that instead of items we are recommending other users to the active user. Although this places extra requirements on the CF framework (described in Section IV), the learning method is sufficiently general to apply to the more standard item recommendation. To see the application to item recommendation, replace in the text occurrences of similar user with similar item, and recommended user with recommended item.

In CF, the rating is calculated based on the rating from similar users, i.e., the value of an unknown rating $r_{c,s}$ for user $c$ and item $s$ is usually computed as an aggregate of the ratings of similar users for the same item $s$ in a user-based CF. In the simplest case, the aggregate can be a simple average. However, the most common aggregation approach is to use a weighted sum based on similarity defined as a distance measure. Since various memory-based algorithms differ in how they define the weights, weight determination becomes a core problem in memory-based CF. Existing work uses simple heuristics to calculate similarities which will be then used as the weights. For example, the correlation approach [6], [8] induces a global model of similarities between users, rather than separate models for classes of ratings.

In this paper, we propose to learn a model for the weights rather than use heuristic similarity. More specifically, the performance of CF relies on how reliable the rating of the similar users is. We define the weight as the reliability of similar users. Thus, the reliability is the contribution of a similar user towards the true rating of the active user in the data rather than the similarity calculated from the rating. We use similarity as a criterion to select a set of users (items in item-based CF), whose rating will be considered in calculating the rating. Furthermore, how those selected users (items in item-based CF) contribute to the rating is personalised for each active user.
Definition For an active user $u_a$, the predicted rating is
\[ o_{a,r} \equiv f(\vec{d}) = \sum_{k \in S} (\omega_k x_k) \] (1)
where $\omega_k$ is the weight of the $k$th similar user, $x_k$ the rating of that similar user and $S$ is the set of similar users.

B. CollabNet: Learning the Contributions

1) Pairwise Training: To learn the contribution, we employ a pairwise approach, i.e., we take pairs of recommended users as instances in learning, and formalise the problem of learning as that of classification. Specifically, in learning, we collect recommended user pairs from the recommendation lists, and for each pair we assign a label representing the order of preference of the two recommended users (items). We then define a loss function and tune the weight values to minimise the loss function.

Without loss of generality, we generate pairs containing as the first element a successful recommended user or item, i.e., a recommended user (item) that has a successful interaction with (selection by) the active user, and the second an unsuccessful recommended one. We always assign value 1 to the label to indicate that the first element in the pair is preferred to the second one. Alternatively, to generate a training pair with the first element an unsuccessful recommended user or item and second a successful one, the label assigned will be 0 instead.

2) Finding Interaction Pairs: The first step towards learning is to find recommended users with ground truth, i.e., with interactions with the active user. Specifically, we find a set of users by CF who have replied positively or negatively to a communication from the active user. An example is shown in Table I, where $u_a$ is the active user and $u_1$ to $u_5$ are users who have had interactions with $u_a$.

<table>
<thead>
<tr>
<th>Table I</th>
<th>USER INTERACTIONS FOR $u_a$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$u_1$</td>
</tr>
<tr>
<td>$u_a$</td>
<td>+</td>
</tr>
</tbody>
</table>

The training instances with respect to $u_a$ can then be formed by pairing a positive user with a negative user in the user interaction table. In the case of Table I, we have pairs $(u_1, u_4)$, $(u_1, u_5)$, $(u_2, u_4)$, $(u_2, u_5)$, $(u_3, u_4)$ and $(u_3, u_5)$, all with label value 1, as training instances.

3) Establishing the Similarity Graph: Since the aim of learning is to find the proper weight values $\omega$ in (1), we need to find the similar users to each of the users in the pair of a training instance. For the example in Table I, a similarity graph can be represented by Table II for item-based CF (where an item is a user in people to people recommendation in social networks).

In Table II, $u_{1,2,3}$ are users who have had positive interactions and $u_{4,5}$ negative interactions with $u_a$ according to Table I. Similar users $u_{1,2,3}$ are also users in training pairs for $u_a$ while users $u_{6,7}$ are other similar users not in a training pair for $u_a$. Similar users are collected based on the similarity definition in Section II. In Table II, a “1” in a cell indicates the users represented by the row and the column of that cell are similar, while a “-” indicates not similar. We have a set of similar users $\{u_2, u_3, u_6\}$ for recommended user $u_1$, for which we need to learn the weight values.

As shown in [17], this is a general formulation where the pairs of ranks need not be complete, i.e., they need not specify a complete ranking of the training data, or even be consistent. As in (1), $f : R^d \rightarrow R$ indicates that the rating of an active user to a test user is specified by the real values that $f$ takes. If the active user prefers user $u_i$ to $u_j$ denoted by $x_i > x_j$, then there exists $f(x_i) > f(x_j)$.

In order to learn the weight $\omega$, a cost function must be defined. Following the approach in [17], we define the probability that the active user prefers $u_i$ to $u_j$ as $P_{ij}(x_i > x_j)$ and the probability of the preference in ground truth as $P_{ij}(x_i > x_j)$, which can then be modelled by a logistic function: $P_{ij} \equiv \frac{e^{o_{ij}}}{1+e^{o_{ij}}}$, where $o_{ij} = f(x_i) - f(x_j)$.

Once we have these two probabilities, the cost function is defined as the difference between those two probabilities, represented by cross entropy:

$$ C_{ij} \equiv C(o_{ij}) = -P_{ij} \log P_{ij} - (1 - P_{ij}) \log (1 - P_{ij}) $$ (2)

Having the cost function defined, the rest is to implement the learning framework. A two-layer neural net trained by back-propagation was used in [17]. However, in CF the large number of users makes the dimensionality $d$ too large for this to be practical. We use stochastic gradient descent to find the optimal weight values for the rating prediction. Thus, we differentiate (2) with respect to the weights: $\frac{dC_{ij}}{d\omega} = \frac{dC_{ij}}{do_{ij}} \frac{do_{ij}}{d\omega} = (P_{ij} - P_{ij})$. Therefore, the weight values are changed in a direction that will reduce the error:

$$ \Delta \omega = -\eta \frac{dC_{ij}}{d\omega} = -\eta (P_{ij} - P_{ij}) \frac{do_{ij}}{d\omega} $$ (3)

where $\eta$ is the positive learning rate that adjusts the relative size of the change in weights. The learning algorithm iteratively changes the weight values according to (3) to reduce the error for each training pair as in Algorithm 1.
Algorithm 1 CollabNet: Learning Collaborative Filtering

Initialise \( n, \omega, \theta, \eta, r \leftarrow 0 \);
repeat
\[ r \leftarrow r + 1; \quad m \leftarrow 0; \quad \Delta \omega_k \leftarrow 0; \]
repeat
\[ m \leftarrow m + 1; \]
\[ \alpha_i \leftarrow \sum_{k \in S} (\omega_k x_i^k); \quad \alpha_j \leftarrow \sum_{k \in S} (\omega_k x_j^k); \]
\[ \Delta \omega_k \leftarrow \Delta \omega_k + \eta (- (P_{ij} - \bar{P}_{ij}) \frac{\partial o_{ij}}{\partial \omega_k}) \]
until \( m = n \)
\[ \omega_k \leftarrow \omega_k + \Delta \omega_k \]
until \( \| \nabla C < \theta \| \)
return \( \omega \);
active user. To generate these pairs, we take a target user from a positive interaction initiated by the active user as the preferred user and another target user from a negative interaction as the second user in the pair. Then the rating for the preferred user is an indicator with value 1 and the second user value 0.

We compare CollabNet with a number of methods including the standard user-based CF [8] and the SocialCollab [18] recommendation methods for people using the evaluation metrics defined in Table VII.

B. Results of Recommendation

As shown in Table VI, SocialCollab achieves about 0.35 SR on average for the top 100 recommendations for each active user. This means an SRI of about 1.25. However, CollabNet boosts the SR to about 0.44 for the top 100 recommendations on average with the highest SR of 0.54 for the top 10 recommendations.

We also compare CollabNet to the standard CF and its extended version CF+ modelling user selection by either initiating an interaction or giving a positive response to a contact initiated by another user [18]. The details of the comparison results of those algorithms on Top 100 and Top 10 are shown in Table VIII, which shows that the proposed algorithm CollabNet outperforms SocialCollab, CF+ and user-based CF for recommendation.

<table>
<thead>
<tr>
<th>SR</th>
<th>Top 100</th>
<th>Top 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>CollabNet</td>
<td>0.44</td>
<td>0.34</td>
</tr>
<tr>
<td>SocialCollab</td>
<td>0.35 (0.09)</td>
<td>0.35 (0.19)</td>
</tr>
<tr>
<td>CF+</td>
<td>0.26 (0.18)</td>
<td>0.30 (0.24)</td>
</tr>
<tr>
<td>CF</td>
<td>0.25 (0.19)</td>
<td>0.28 (0.26)</td>
</tr>
<tr>
<td>Default</td>
<td>0.28 (0.16)</td>
<td>0.28 (0.26)</td>
</tr>
</tbody>
</table>

We also compare CollabNet to the recommendation performance of the other algorithms on the top 10 of the ranked recommendation list as shown in Figure 1. We can see that although without learning SocialCollab performs better than default (DSR) on the top 10 recommendations, CollabNet performs much better than SocialCollab on the top 10 recommendations on both SR and Recall. When we compare CollabNet to CF+ and standard CF on the top 10 recommendations, it shows that standard CF performs at just about the default level and CF+ performs slightly better than standard CF, however, the most significant improvement is achieved by CollabNet, where it significantly outperforms all other methods both of SR and Recall.

C. Results of Ranking

We also show results for ranking in Figure 2, where a point in the positive RI curve denotes an improved ranking of the recommended users for an active user by learning, while that in the negative RI curve denotes a worse ranking. In Figure 2, the positive RI curve covering a large range of active users indicates that there are many more active users receiving a better ranking by learning. The area under the RI curve (AURIC) for positive RI is about 3 times that for negative RI, which demonstrates that the increase of RI is much higher than the decrease of RI for the same number of active users. This can also be seen in Figure 2 where cumulative ranking improvement (CRI) is shown, which indicates the stable increase in RI with the increase in the number of active users.

VI. CONCLUDING REMARKS

We have proposed an approach for people recommendation by collaborative filtering and machine learning. Our experimental results show that the novel CollabNet recommender performs well in people to people recommendation on data from a commercial online social networking site. The proposed learning algorithm is able to rank the recommendations in order to further improve the success of predicted user interactions. The proposed algorithm CollabNet outperforms all other methods including standard CF as measured on both Precision (SR) and Recall. For future work, we will investigate the scalability of CollabNet, the
“cold start” problem of providing recommendations for new users, and a general probabilistic framework for ranking in the context of the CollabNet algorithm, as well as compare CollabNet with approaches to link prediction.

ACKNOWLEDGMENT
This project is funded by the Smart Services Cooperative Research Centre.

REFERENCES

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Success Rate (SR) or Precision</td>
<td>The proportion of the true predicted successful interactions (PSI) to all predicted successful interactions: $SR = \frac{\text{TPSI}}{\text{APSI}}$, where $\text{TPSI}$ is the number of true PSI and $\text{APSI}$ the number of PSI.</td>
</tr>
<tr>
<td>Default Success Rate (DSR)</td>
<td>The proportion of successful interactions (SI) to all interactions in the dataset: $\text{DSR} = \frac{\text{SI}}{\text{All}}$, where $\text{SI}$ is the number of true SI and $\text{All}$ the number of all interactions.</td>
</tr>
<tr>
<td>Success Rate Improvement (SRI)</td>
<td>The ratio of success rate to the default success rate: $\text{SRI} = \frac{\text{SR}}{\text{DSR}}$, where $\text{SR}$ is the number of all interactions.</td>
</tr>
<tr>
<td>Recall</td>
<td>The proportion of the true PSI to all true SI: $\text{Recall} = \frac{\text{TPSI}}{\text{All}}$.</td>
</tr>
<tr>
<td>Ranking Improvement (RI)</td>
<td>The product of the difference between the predicted rank and ground truth rank and the response indicator: $\text{RI} = \sum_1^n r_{ij} = (n g_j - n p_j) \times s_j$, where $g_j$ is the reference rank index for recommended user $j$, $p_j$ the predicted rank index, $n_j$ the number of recommended users for the active user $i$. Positive RI values indicate improved ranking.</td>
</tr>
</tbody>
</table>

Table VIII

<table>
<thead>
<tr>
<th>Name</th>
<th>Top 10</th>
<th>Top 20</th>
<th>Top 30</th>
<th>Top 40</th>
<th>Top 50</th>
<th>Top 60</th>
<th>Top 70</th>
<th>Top 80</th>
<th>Top 90</th>
<th>Top 100</th>
<th>Top 200</th>
<th>Top 300</th>
<th>Top 400</th>
<th>Top 500</th>
<th>Top 600</th>
<th>Top 700</th>
<th>Top 800</th>
<th>Top 900</th>
<th>Top 1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>CollabNet</td>
<td>0.54</td>
<td>0.50</td>
<td>0.49</td>
<td>0.47</td>
<td>0.47</td>
<td>0.45</td>
<td>0.45</td>
<td>0.44</td>
<td>0.44</td>
<td>0.40</td>
<td>0.39</td>
<td>0.38</td>
<td>0.37</td>
<td>0.37</td>
<td>0.37</td>
<td>0.37</td>
<td>0.37</td>
<td>0.37</td>
<td></td>
</tr>
<tr>
<td>SocialCollab</td>
<td>0.35</td>
<td>0.34</td>
<td>0.34</td>
<td>0.34</td>
<td>0.35</td>
<td>0.35</td>
<td>0.35</td>
<td>0.35</td>
<td>0.35</td>
<td>0.35</td>
<td>0.36</td>
<td>0.36</td>
<td>0.37</td>
<td>0.37</td>
<td>0.37</td>
<td>0.37</td>
<td>0.37</td>
<td>0.37</td>
<td></td>
</tr>
<tr>
<td>CF+</td>
<td>0.30</td>
<td>0.28</td>
<td>0.27</td>
<td>0.27</td>
<td>0.28</td>
<td>0.28</td>
<td>0.27</td>
<td>0.27</td>
<td>0.26</td>
<td>0.26</td>
<td>0.27</td>
<td>0.27</td>
<td>0.28</td>
<td>0.28</td>
<td>0.28</td>
<td>0.28</td>
<td>0.28</td>
<td>0.28</td>
<td></td>
</tr>
<tr>
<td>CF</td>
<td>0.28</td>
<td>0.27</td>
<td>0.26</td>
<td>0.26</td>
<td>0.26</td>
<td>0.26</td>
<td>0.26</td>
<td>0.25</td>
<td>0.26</td>
<td>0.26</td>
<td>0.26</td>
<td>0.27</td>
<td>0.27</td>
<td>0.27</td>
<td>0.27</td>
<td>0.27</td>
<td>0.27</td>
<td>0.27</td>
<td></td>
</tr>
</tbody>
</table>