Open Domain Recommendation: Social networks and collaborative filtering

Sarah K Tyler and Yi Zhang

University of California, Santa Cruz

Abstract. Commercial enterprises employee data mining techniques to recommend products to their customers. This recommendation is based on similar purchasing histories, as well as similarities between users and similarities between products. It typically involves analyzing a specific domain such as movies or books to make predictions as to which user is likely to want which product, increasing sales and providing users with desirable content. But when the domain isn't as specific, or the data becomes too sparse to draw similarities between users, prediction becomes much more difficult. One avenue to find similarity in users is to utilize the user provided social connections, under the assumption they share some like-mindedness. We propose using social network data, along with the contextual information to enhance product recommendations in a sparse context. We show that when a social network can be applied, it is a strong indicator of user preference for product recommendations, having high precision at the cost of recall. Although the sparseness of the data may mean that the social network is not always applicable, we present a solution to utilize the network in these cases.

1 Introduction

Recommendation systems use features from both users and products to predict which products a given user might like, typically operating over a specific domain. A classic example is movie recommendation with either the MovieLens or Netflix data. While the number of movies in general may be large, the number of new movies produced each year is relatively small, and the number of blockbusters seen by large audiences is even smaller. Therefore, for large audiences there is often a lot of overlap between the movies any two individuals has seen, resulting in a rating prediction with a higher degree of accuracy. If two users have both rated the same subset of movies, one can aggregate the information to see where their similarities and differences lie; if they share similar genre tastes, favorite actors or other aspects in common. If the number of movies they have both rated is relatively small, it is harder to determine the underlying causes that makes the ratings either similar or dissimilar with respect to each other. In these contexts, additional information is necessary.

By looking at additional contextual information such as product descriptions as well as social and trust networks, we hope to compensate for the data sparseness. If the user has selected a group of friends, it is reasonable to assume these friends share some interests in common. While a user may add anyone to their

social network, there is an implicit assumption of kinship: work acquaintance might share a professional interest while social friends may share hobbies. While the connection between any two individuals may be tenuous, a large group of connections can potentially bias the recommendation system in the right direction. Our goal is to combine these multiple types of information to enhance prediction when the small number of training ratings per product hamper the accuracy of traditional methods.

2 Related Work

Typically recommendation filtering uses collaborative filtering, such as singular value decomposition[1], since it scale well and account for individual's uniqueness. The idea is to find similar individuals, and similar products, in order to make the best recommendation. The drawback to these approaches is that they require a lot of data per user and per product.

Content based approaches such as latent semantic indexing[5], also help improve recommendation[9], but face scaling issues. As new items are introduced, they may have additional, never before seen features. This is especially true when the context includes natural language such as product description terms, where, according to Zipf's law we can expect to see new words, regardless of how large our training set is.

The notion of using trust networks is not new, both with implicit and explicit networks. Referral Chains use word co-occurrences to build an informal network and uses the network to enhance web search[6]. Yet the question seems to be how to apply it since social connections are not necessarily the same as trusted connections, and may be tenuous at best.[8]

TidalTrust is a recommendation system that allows users to specify a degree of trust for each person in the network.[4][3] The trust is then accumulated over neighbors of varying distance to create a ranked list approach. The top ranked items are then presented to the user, and are the only ones used in the evaluation. In fact, the trust network approach has been discussed in conjunction with the same data set we use. Massa et al.[7] showed the potential power of trust networks, and argued for a way of propagating trust over all users.

3 Methods

For our experiments we looked at three types of methods: a base line, a social network based approach and a collaborative filtering approach that is more commonly used for product recommendation in the movie domain. As you will see, the social network method worked the best in terms of precision, but was applicable in a limited context thus having low recall. The collaborative filtering, on the other hand, fell victim to the sparcitivity of the data. Being a more complex model, it did not have sufficient data to learn the hidden representations and was not able to outperform the baseline.

3.1 Baseline

In order to establish whether personalization provided any improvements, we first needed a baseline that predicted each product rating independent of the users and sans personalization. This baseline predicted a product's rating by taking the average rating for the given product in the training set. One added benefit of this approach is that it is very fast, as there are no hidden features to estimate. However, we predict it will not do well since (1) the data is sparse and can easily be skewed by a single review and (2) some individuals have a predisposition to weight products either high or low and this model makes no account for that. The approach is detailed in Equation 1.

$$r_i = \frac{1}{n_i} \Sigma_u r_{iu} \tag{1}$$

Here n_i is the number of users that have a non missing rating for product i, $n_i = ||\{u \; where \; r_{iu} \; is \; defined\}||$. The rating of product i is the average of user u's ratings of product i, r_{ui}.

3.2 TidalTrust

We used TidalTrust, as our social network recommendation system since it provides a way of propagating trust as Massa et al.[7] called for. TidalTrust utilizes the extended trust network for users by creating a trust flow metric. As two users grow further apart, the amount of trust between them is the product of the trust along each part of the path. In cases where there is more than one path, only the path with the highest trust score is used. When a prediction is needed for a specific user and product, the trust network is explored to find other users who rated the same product. If a member of the trust network has rated the item, their opinion is weighted according to their distance from the initial user, and by how much trust is exhibited along the path between the two users. The approach is outlined in Equation 2.

$$r_{iu_1} = \frac{\Sigma_{u_2 \in S} t_{u_1 u_2} r_{iu_2}}{\Sigma_{u_2 \in S} t_{u_1 u_2}} \tag{2}$$

Here the rating of product i for user u is the sum of the trust of u_1 to u_2, $t_{u_1 u_2}$ multiplied by the rating of product i by user u_2, r_{iu_2}, divided by the total sum of the trust between user u_1 and all users in the trust network.

In the initial TidalTrust paper, users could specify the degree to which they trusted another user. In our data set, however, all we can see is whether or not a user trusts another user. There is no notion of how or why users trust each other so we are forced to assume uniform trust in our implementation. Thus our implementation is a weighted average according to distance.

3.3 Singular Value Decomposition

The collaborative filtering approach we used was singular value decomposition [1] which reduces the dimensionality of both user features and product features by representing users as a product vector where each index in the vector represents the product's rating, and products as a user vector where each index represents the user's rating of the product.

In our case, the singular value decomposition will likely be less effective because of how simultaneously diverse and how sparse our data is. When dimensionality of the vectors is high, each item begins to look equally far away. For our data, we varied the number of hidden features. Since our data is limited, it would be difficult to learn many features, however our diverse domain indicates more features would be necessary.

4 Data

The data set used was collected was from Epinions.com[2] in January of 2008. Epinions.com is a commercial site that allows users to rate products, give reviews, and share their opinions with other users, offering several unique challenges.

One of the advantages of Epinions compared to other review sites is that users can rate any kind of item they want, from mattresses to restaurants. While a benefit to the epinions.com user base, this increases the size of the domain to make predictions about. As a result, the number of hidden features required to make an accurate prediction is likely greater than in domains where there is more consistency between products.

Second, the same feature value may have very different effects on two different products. For example, a $100 laptop may seem like a steal, but a $100 pack of gum is likely over priced. Here the same feature value for price influences the recommendation in two different ways even for the same user.

The data sample itself consists of 85,139 user-product ratings, each on a scale of 1-5. The average user-product rating was 4.02 with a standard deviation of 1.14. There are 35,137 unique products and 41,696 unique users. This means each user has an average of two reviews, and each product has on average 2.4 ratings.

Figure **??** shows the break down of training examples per test example. As you can see the data follows an approximately Zipf law curve. What's not seen in Figure **??** is the 7,428 products in the test set with zero reviews in the training set. This is compared to the 14,875 total test cases. A 80/20 split was used for training and testing, randomly sampled.

5 Evaluation

In order to evaluate the different models, we needed to establish some objectives. In the first case we wanted to see how closely the model confirms to grown

[1] We acquired our implementation of Singular Value Decomposition from TimelyDevelopment[1] which they used for the Netflix Prize and distribute freely.[10].

truth and the objective was to minimize the error. However, finding the grown truth value is not the primary goal of recommender systems. When creating a recommender system the primary task is to ensure the item we recommend is desirable to the user and not necessarily the degree to which it is desirable or not desirable. Therefore, in the second case, our objective is not how closely the model represents grown truth, but whether the highly rated products are desirable to the end users.

The data follows a roughly Gaussian distribution, with a center of 4.02. Therefore, to achieve the first objective we used mean squared error (MSE) which fallows naturally from the distribution. For the second objective we used several different methods. The first was a precision recall curve for various "desirable" thresholds. For example, if the "desirable" threshold was 4, we assumed a user who rated the product with a 4 or a 5 liked the given product, but a user who rated it with a 1, 2 or 3 did not. In Recommender systems, precision is more important than recall, so our discussion often centers on precision.

Often users will only look at the first couple of items. To measure how well our models would perform in this environment, we also include macro averages Precision@K and AverageRating@K. It turns out in our environment the number of reviews per product is often low, following a long tail distribution. In half the instances in our testing subset, the user only had one item. Since K then becomes 1, we also looked at an aggregate Precision@K that was independent of users. The following subsections discuss our results. For this evaluation, we selected only two thresholds for "desirable" or user preference when computing these measures. The first, 4.0 reflects the average rating users tend towards. Using the average tends to lead to very good results for all models, so we also choose a threshold of 4.5. Since users are bound by integer ratings, this means only products that received a 5/5 were deemed interesting, and the models needed to make a prediction that rounded to 5.

Our original 20% split of our 80/20 partitioning consisted of 14,873 test cases. However, only 7,445 contained a product or user also in the 80% partition used for testing. We used this subset of our testing examples for the following evaluations.

5.1 Baseline

For our first objective, we calculated a MSE of 1.36. The F-measure for the baseline in Figure **??** exhibits a knee at 3.5. This is partially because the average measurement calculated in the baseline can be skewed by either a very strong or very negative rating. The closer the threshold for "desirable" is to the maximum value or minimum value, the less likely it is for the average rating to recover from a strongly dissenting rating, especially when half of the data that occurs in the test set occurs less than twice. This is also reflective in the sharp downtick of precision, recall and F-measure around 4.5.

Coincidently, the average rating is 4.02 with a standard deviation of 1.14. The high precision, recall and F-measure for the "desirable" threshold of less than 3 are inflated since the data is so strongly skewed.

The AverageRating@K for $k = 1$ was 4.0 and the AveragePrecision@K for $k = 1$ item was 0.84 and 0.59 for the 4.0 and 4.5 thresholds of "desirability" respectively. This shows that the baseline is reasonably accurate in predicting whether an item is in the top half of all items, but is not as accurate at predicting the very top rated items.

5.2 TidalTrust

The TidalTrust MSE was extremely low, 0.52, less than half the MSE for the baseline although we only calculated the MSE in products were prediction was possible. Along similar lines, precision was very high as shown in Figure **??**. Recall, however, was less than 0.008 for all "desirable" thresholds. This was an artifact of how few instances could be rated with TidalTrust. While TidalTrust was able to make strong predictions, it was only able to make a prediction in 75 of the 7,445 test cases, or 1% of the time. It appears that if a product was rated by someone in the given trust network, the two users often agree both in polarity and in magnitude of the rating. However, most of the time two users in a given trust network do not rate the same product. In our data set, a given trust network generally may only have one item that exists in common with more than one user. Still, this approach shows promise.

The strong MSE metric in general along with the precision curve show that trust networks do have a positive effect on rating prediction, but more work needs to be done to harness that power.

The baseline, when run over the 75 instances that could be predicted using TidalTrust, had a MSE of 0.49, very close and on the surface have slightly better than the MSE of TidalTrust. However, in those 75 instances, TidalTrust exactly agreed with the true rating value in 59 instances. In contrast, the baseline was within 0.25 of the true rating in only 32 instances. If we increase the window to 0.5 the baseline was within the true value in 38 instances. When the trust network could make a prediction, it seems to be more accurate than the baseline at finding the exact rating.

In fact, TidalTrust so closely matched the predicted value, that it was only off from the actual value by more than 1 in three of the seventy five instances. When it was off, however, the difference was significant. In one instance TidalTrust's predicted rating was off by 4 from the true rating, and in another it was off by 3. In contrast, the greatest the baseline was off on these 75 instances was 2.2, however there were 10 instances where the baseline was off by more than 1 and less than 2 to the true rating. If you treat the three instances of magnitude greater than 2 as outliers from both the baseline and TidalTrust, the MSE of TidalTrust is 0.23 where the MSE of the baseline is still 0.48. While in general MSE may not be an appropriate measure for recommendation accuracy, the measure does support that Tidal Trust, on average, beats the baseline.

Figure **??** and **??** offer a different perspective of the same phenomenon with the Precision@K metric. Here, items are ranked according to their predicted rating, independent of users. We see that the precision values are near 1 for all

values of K. In environments were precision far outweighs recall, such as product recommendation, there is a clear advantage for Tidal Trust.

AverageRating@K for $k = 1$ was 4.1. The average precision rating for the top 1 item per users are also reflective of the same nature, 96.7 and 93.4 for the 4.0 and 4.5 thresholds respectively. This shows that Social Network information consistently influence product recommendations when they can be applied.

5.3 Singular Value Decomposition

The final approach used was collaborative filtering in the form of singular value decomposition.

In this case the MSE dropped considerably to 1.5, just above the baseline. This is indicative of two things. The data itself is so sparse that the hidden features cannot be trained well, and that collaborative filtering may require much more data than is available.

The precision and recall curves show something are similar to the curves in Figure **??**, and are nearly identical to the baseline. We see, however that Percision is slightly higher than the baseline by Figures **??** and **??** according to the Precision@K metric.

In terms of the macro average, Precision@K for k of 1, SVD scores 86.2 62.7 for thresholds of 4.0 and 4.5 respectively. Like the baseline, the SVD model is more accurate when it predicts a product is in the top 50

MovieLens Data It is worth noting that Singular Value Decomposition is the collaborative filtering technique that is often used in the movie domain because it is able to distill the movies and users into smaller feature vectors where better inference can be drawn over them.

The key difference between MovieLens and Epinions is the amount of reviews per user and product. In MovieLens data each user has an average of 133 reviews and the products have 222 reviews. This additional information provides more certainty when learning the hidden features.

Our Epinions.com data is simultaneously very sparse, making it difficult to learn hidden features, and very vast, making a greater number of hidden features necessary. This makes sense since we have, on average, 2.4 ratings per product and many products have 1 rating. Learning multiple hidden features is difficult with the limited amount of data.

6 Conclusions

It is clear that the dimensionality requirements of Singular Value Decomposition makes the model complexity too high in the epinions.com context. Although Singular value decomposition outperforms the baseline, it does not out perform it by much. This indicates it was unable to adequately learn the hidden features and improve recommendation accuracy.

The trust network proved to be very useful in the limited number of times it could be applied. It's accuracy was higher than the baseline for both evaluation objectives. Even without considering the possible outliers, the trust network preformed very well. It seems when a product is reviewed by your trust network, your opinion of it will be closer to that of your network, rather than the entire user base. Unfortunately, it could rarely be applied, limiting it's utility. In our sample it was only able to make a prediction less than 1% of the time. If the social network information could be applied in more cases, the accuracy of the recommendation would surely increase.

7 Future Work

The social network aspect of the data shows great promise for improving recommendation, but the current system is not capable of capitalizing on it. If one can bootstrap the trust network, they may see a significant improvement.

While a specific product may not be common in a trust network, the trust network may share general interests, which then influences what products they would like to purchase. Thus the trust network's relative opinions of similar products may be useful for rating prediction. For example, a Single Lens Reflex (SLR) is a type of advanced lens for digital cameras. For the average user, the additional cost may be prohibitive for them to purchase a camera with such a lens, but to an amateur photographer, the lens may be worthwhile. It is likely that our amateur photographer has friends, colleagues or acquaintances with whom he or she shares his or her interests. If these acquaintances are in the social network of our amateur photographer, they may rate similar products with SLR lenses. Therefore, other products with similar features may have an average rating in the social network that is close to our amateur photographers' true rating for the given product. This can help us estimate the rating for a new, unseen product from the explicit network in the same way that collaborative filtering is often applied for the implicit network.

We propose using the product contextual features to find similar products. The social network recommendation for each product can be weighted according to the similarity of the desired product in addition to the trust distance between user and network. This approximate rating can be used in cases where the predicted rating cannot be established directly. It is our belief that this method will show continued improvement over collaborative filtering in this general domain.

References

1. T. Development. Timely development singular value decomposition inplementation. http://www.timelydevelopment.com/Demos/NetflixPrize.htm (visited on Jan. 20, 2008), 2008.
2. Epinions.com. Epinions.com. http://www.epinions.com (visited on Jan. 20, 2008), 2008.

3. J. Golbeck. Personalizing applications through integration of inferred trust values in semantic web-based social networks. In *Proceedings of Semantic Network Analysis Workshop*, 2005.
4. J. Golbeck. Generating predictive movie recommendations from trust in social networks. In *Proceedings of the Fourth International Conference on Trust Management*, 2006.
5. T. Hofmann. Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst.*, 22(1):89–115, 2004.
6. H. Kautz, B. Selman, and M. Shah. Referral web: combining social networks and collaborative filtering. *Commun. ACM*, 40(3):63–65, 1997.
7. P. Massa and B. Bhattacharjee. Using trust in recommender systems: An experimental analysis. In *Trust Management: Second International Conference, ITrust 2004, Oxford, UK, March/April 2004 Proceedings*, 2004.
8. D. W. McDonald. Recommending collaboration with social networks: a comparative evaluation. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, 2003.
9. P. Melville, R. J. Mooney, and R. Nagarajan. Content-boosted collaborative filtering for improved recommendations. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-2002)*, Edmonton, Canada, 2002.
10. Netflix. Netflix prize. http://www.netflixprize.com (visited on Nov. 30, 2006), 2006.

References

Clarke, F., Ekeland, I.: Nonlinear oscillations and boundary-value problems for Hamiltonian systems. Arch. Rat. Mech. Anal. **78** (1982) 315–333

This article was processed using the LaTeX macro package with LLNCS style