Clustering Items for Collaborative Filtering

Mark O'Connor & Jon Herlocker
Dept. of Computer Science and Engineering
University of Minnesota
Minneapolis, MN
{oconnor,herlocke}@cs.umn.edu

ABSTRACT

This short paper reports on work in progress related to applying data partitioning/clustering algorithms to ratings data in collaborative filtering. We use existing data partitioning and clustering algorithms to partition the set of items based on user rating data. Predictions are then computed independently within each partition. Ideally, partitioning will improve the quality of collaborative filtering predictions and increase the scalability of collaborative filtering systems. We report preliminary results that suggest that partitioning algorithms can greatly increase scalability, but we have mixed results on improving accuracy. However, partitioning based on ratings data does result in more accurate predictions than random partitioning, and the results are similar to those when the data is partitioned based on a known content classification.

Keywords

Collaborative filtering, partitioning, clustering, GroupLens, MovieLens.

1. INTRODUCTION

Recommender systems based on automated collaborative filtering predict new items of interest for a user based on predictive relationships discovered between that user and other participants of a community. Most of the successful research and commercial systems in collaborative filtering use a nearest-neighbor model for generating predictions. Automated collaborative filtering systems based on the nearest-neighbor method work in three simple phases:

1. Users of an automated collaborative filtering system rate items that they have previously experienced.
2. The automated collaborative filtering system matches the user with other participants of the system who have similar rating patterns (i.e. they have similar opinions on experienced items.) This is usually done through statistical correlation. The closest matches are selected, becoming known as *neighbors* of the user, or collectively as the *neighborhood*.
3. Items that the neighbors have experienced and rated highly, but which the user has not yet experienced, will be recommended to the user, ranked based on the closeness of the neighbors to the user and the consistency of opinion within the neighborhood.

Automated collaborative filtering systems are being applied to larger and larger sets of items. With large numbers of items in the prediction domain, we see the occurrence of three significant negative phenomena:

1. Since users have limited resources to experience items (read articles, see movies, listen to music), the density of user ratings on items decreases. It becomes less likely that any significant number of a user's neighbors will have experienced the item for which a prediction is being requested
2. While the density will decrease, the number of items that must be considered in each user to user correlation will still increase, increasing the amount of time necessary to compute the neighborhood.
3. As the number of items in the prediction domain gets large, the diversity of those items will also increase (otherwise, why would you need to recommender system to select between them?). As this diversity increases, it becomes less likely that a user's opinions on all other items will be relevant to his opinion on a single given item.

As a result of these negative phenomena, it becomes hard to scale automated collaborative filtering systems to large numbers of items while maintaining reasonable prediction performance.

The GroupLens automated collaborative filtering system for Usenet news[7] was the first automated collaborative filtering system to deal with massive item sets. GroupLens addressed the large item set issue by creating a separate item partition for each Usenet discussion group. A user's prediction within a newsgroup such as rec.humor was computed using only ratings of other messages within rec.humor. Thus the prediction was only influenced by opinions on humor, and not by opinions on non-related topics, such as cooking (rec.food.recipes). Since each reader of a newsgroup was presented with the same set of articles to read, the ratings were denser, and there was more overlap between users' ratings.

We were able to accomplish this with Usenet news, because of the existing well-defined partitioning of news articles based on content. However, we would like a partitioning scheme that can be generally applied to any set of items regardless of the content. This would relieve us from having to design a new contentspecific partitioning technique for every new type of item we predict.

Our approach is to partition items based on user rating data that was collected for the purpose of collaborative filtering, which will exist no matter what type of content we are recommending. We believe that we can use this rating data to discover relationships between items that would allow us to partition the items effectively for the purpose of improving accuracy and performance in automated collaborative filtering systems.

We are currently exploring the use of existing data clustering and partitioning techniques [4] to handle partitioning items sets based only on user ratings for those items.

2. PARTITIONING ITEMS IN COLLABORATIVE FILTERING

Partitioning the item-space reduces one large-dimensionality space into a set of smaller-dimensionality spaces; with fewer items, less ratings, and often less users. Once this space has been partitioned, we apply the traditional collaborative filtering algorithms[3, 7, 8] within each partition, independent of the other partitions. The time to compute a prediction will decrease, since there is less data to consider and the density of ratings should increase because people who consume one item within a cluster are more likely to have read similar items that occur within the partition. We also hope that by clustering together similar items within each partition, prediction accuracy will increase because we have removed the noise generated by ratings on items of dissimilar content or user interest.

Most clustering and partitioning algorithms require a distance metric or similarity metric to guide the clustering process. In order to provide this metric we need to compute a similarity between items. The measure we used to calculate similarity between items was the Pearson correlation coefficient. In effect, we are computing the extent to which two items are similarly rated by users. Intuitively, two movies will have a high correlation if in general, users felt the same way about both movies. We did not consider the effect of negative correlations, because the partitioning algorithms considered were not designed to handle negative similarities.

3. ALGORITHMS

Hundreds of variants of clustering and partitioning algorithms have been published. In order to begin exploring the potential of partitioning algorithms for items in collaborative filtering, we have chosen to experiment with several well-known clustering/partitioning algorithms with easily available implementations.

We chose to experiment with four algorithms:

- Average link hierarchical agglomerative [1]
- ROCK [2] – A Robust Clustering Algorithm for Categorical Attributes
- kMetis, and hMetis [5, 6] – Multilevel k-way Graph Partitioning

The average link clustering algorithm is one of the classic basic clustering algorithms, and was chosen to supply a base clustering case. ROCK is a recent clustering algorithm developed at Bell Labs and is supposed to have improved performance on categorical data such as used in the MovieLens dataset. kMetis and hMetis are high-speed graph (and hypergraph) partitioning algorithms developed at the University of Minnesota.

We compared the results of clustering items in rating space to clustering based on genres and also to random clustering. Random clustering provides a baseline against which we can compare clustering algorithm variants. Genres are a well-known content attribute of movies, and using them allows us to compare rating-based clustering to simple content-based clustering.

4. EXPERIMENTAL DATA

We are currently performing experiments on a subset of movie rating data collected from the *MovieLens* web-based recommender (movielens.umn.edu) (this has a similar form to, but is not the same as the well known EachMovie dataset.) The data set used contained 100,000 ratings from 943 users and 1,682 movies, with each user rating at least 20 items. The ratings in the MovieLens data were explicitly entered by users, and are integers ranging from 1 to 5.

Some preprocessing was performed on the dataset, in order to improve conditions for clustering. Any movie that correlated with less than 5 other movies was discarded, along with all its ratings. Also, any movie with less than 10 ratings was also discarded. This narrowed the data set to 1,066 movies. Using the 1,066 movies further narrowed the test set from 9,430 ratings to 9,166.

5. EXPERIMENTAL PROCEDURE

The data was divided into two parts, a training set and a test set. The test set contained 10 ratings from each user, and the training set contained the remaining ratings. A movie versus movie correlation matrix was computed using only the training set. Movies that had less than 10 users in common (i.e. movies for which there are less than 10 people who have rated both) were considered to have a zero correlation, since we did not consider 10 to be enough information to produce a confident correlation.

The movie versus movie matrix is used as input to the each of the partitioning algorithms, resulting in a partitioning of items for each algorithm. The procedure for computing predictions is as follows:

```
for each clustering algorithm
   compute partitions of items
   for each partition
      compute predictions for test set based
      on Nearest Neighbor Collaborative
      Filtering
```

6. EVALUATION

The following criteria was used to evaluate each prediction strategy:

- Mean absolute error (MAE). The mean absolute error between the predicted ratings and the actual ratings of users within the test set.
- Coverage. The percentage of desired ratings that could be predicted by the algorithm.

Method	MAE	Coverage
Unpartitioned base case	0.7594	99
Random	0.8211	74
Genre	0.7806	87
Average Link k = 50	0.7754	61
hMetis k = 5, Ubfactor = 10	0.7859	79
hMetis k = 5, Ubfactor = 1	0.7951	82
kMetis k = 5	0.8033	79

Table 1: Summary of Results. Ubfactor is the hMetis unbalance parameter. k is the number of desired clusters

7. RESULTS

Results from the different experiments are shown in Table 1. The mean absolute error is computed over all partitions generated by a clustering algorithm.

7.1 Average Link

The Average Link algorithm is controlled by one parameter, k which is the number of desired partitions. Low values of k resulted in one mega cluster containing all but k-1 movies. Larger values of k (e.g. k greater than 50) resulted in a more even distribution but still many small useless clusters (see Figure 1). Coverage using Average Link was low due to the inability to predict for items occurring in the small clusters.

7.2 ROCK

The ROCK algorithm is controlled by two parameters: k, which is the desired number of partitions and θ, which is a threshold point that determines what correlations are considered. Results of the ROCK algorithm were worse than those of Average Link. No matter what settings of the parameters k and θ, there was only one cluster with more than one item. Because clustering with ROCK was ineffective we did not attempt to compute predictions within its clusters.

7.3 Metis

Metis algorithms were the only algorithms that provided reasonably distributed clusters. The Metis algorithms attempt to create clusters of the same size. The kMetis algorithm guarantees

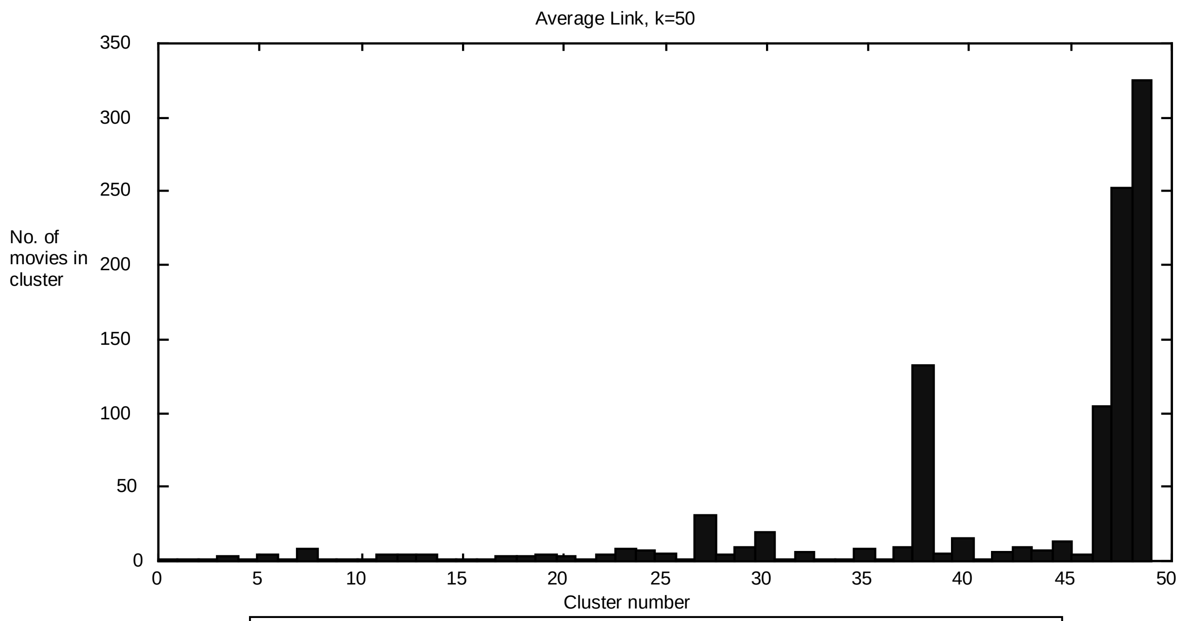

Figure 1: Distribution of movies within clusters generated by the Average Link clustering algorithm when 50 clusters are requested.

all clusters will be approximately equally distributed, while the hMetis has an unbalance parameter (Ubfactor) which allows the algorithm to deviate from an equal distribution. Large values of the unbalance parameter provide partitionings that were similar to those of Average Link and ROCK.

Predictions computed in Metis generated clusters were more accurate than predictions computed in randomly generated clusters. In addition the Metis partitioning was able to produce higher coverage than the random partitioning. Predictions based on Metis generated clusters approximately as accurate as those based on the Genre clustering although coverage was lower than with Genre clustering.

The hMetis algorithm did not provide significantly better results than the kMetis algorithm. The kMetis algorithm is preferable because of its shorter execution time.

8. DISCUSSION

We have three goals when partitioning items in a collaborative filtering system. We want to reduce the amount of computation time, increase the extent in which we can compute predictions in parallel, and increase the accuracy of predictions.

The kMetis graph partitioning algorithm was the most promising clustering algorithm that we experimented with. While accuracy of predictions computed on kMetis generated partitions was not as good as the unpartitioned base case, both accuracy and coverage were better than with the random partitioning. In addition, the results using kMetis were similar to those using genre partitioning. This suggests that an intelligent (non-random) partitioning algorithm in rating space can perform as well as partitioning on known content attributes.

By partitioning the item space into numerous smaller clusters each individual prediction computation will take less time to complete. Since each partition is independent of others, prediction computation on each partition could be done in parallel, further increasing the rate at which predictions can be computed.

One of our hypotheses was that we could increase prediction accuracy by clustering together movies that were similarly rated. However, we did not find this to be the case consistently in our experiments. Only in one or two cases was accuracy for items in a partition greater than the base unpartitioned case. This could be due to the fact that correlation of ratings between two items measures how similarly two items are rated, and not necessarily how similar in content those two items are. It could also be due to the fact that we are restricting items to being exclusively in one cluster. Certain items may have significant predictive value for multiple clusters and removing them may reduce the overall accuracy.

9. FUTURE WORK

This short paper presents some initial results from work currently in progress. We are continuing to explore the effectiveness of clustering and partitioning techniques within rating spaces of collaborative filtering. Further work will continue to explore what characteristics of a clustering or partitioning algorithm make it effective for partitioning items in collaborative filtering.

We are particularly interested in examining algorithms that create clusters with non-exclusive item membership. A simple example of this would be a clustering algorithm that allowed items to occur in multiple clusters, while still minimizing the size of the clusters. More complex algorithms might have an item belonging fractionally to multiple clusters. We believe that this more closely models the “taste space” of the real world, where each item may appeal to a set of different user “tastes”.

10. REFERENCES

[1] E. Gose, R. Johnsonbaugh, and S. Jost. Pattern Recognition and Image Analysis. Prentice Hall, 1996.

[2] S. Guha, R. Rastogi, and K. Shim. ROCK: a robust clustering algorithm for categorical attributes. In Proc. of the 15th Int’l Conf. On Data Eng., 1999.

[3] Hill, W. et al. Recommending and Evaluating Choices in a Virtual Community of Use. In ”. In *Proceedings of ACM CHI’95 Conference on Human Factors in Computing Systems*, pages 194-201. 1995.

[4] A.K. Jain and R.C. Dubes. Algorithms for Clustering Data. Prentice Hall, 1988.

[5] G. Karypis, and V. Kumar. Metis: a software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orders of sparse matrices. Available on WWW at URL: http://www.cs.umn.edu/~karypis/metis.

[6] G. Karypis, and V. Kumar. hMetis: a hypergraph partitioning package. Available on WWW at URL: http://www.cs.umn.edu/~karypis/metis/hmetis.

[7] Resnick, P., et al. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. *In Proceedings of ACM CSCW’94 Conference on Computer-Supported Cooperative Work*, pages 175—186. 1994.

[8] Shardanand, U., and Maes, P. Social Information Filtering: Algorithms for Automating “Word of Mouth”. In *Proceedings of ACM CHI’95 Conference on Human Factors in Computing Systems*, pages 210-217. 1995.