Progress Report: Collaborative Filtering Using Bregman Co-clustering

Wei Tang, Srivatsan Ramanujam, and Andrew Dreher

April 4, 2008

1 Introduction

Analytics are becoming increasingly important for business competition. Many of the top brands today employ analytics to improve efficiency and sales while at the same time maintaining increased customer satisfaction and loyalty. From a consumer's point of view, these analytical systems are often manifested in the form of recommender systems, which use accrued user data to predict the items that would be of the most interest to the user. As larger selections are made available to users, often through the power of e-commerce, recommender systems are an increasingly important part of the user experience, and if they are executed well, such systems can increase sales and generate strong user loyalty.

With so much emphasis being placed on these analytic tools, increased performance, both in terms of accuracy and computational efficiency, is required. In our course project, we focus on the strategy known as collaborative filtering. Collaborative filtering recommender systems often utilize unsupervised learning algorithms, and since the data often naturally occurs in the form of co-occurrence tables, a popular, classic method is one-way clustering, in which one dimension of the table, such as the rows, is grouped at a time according to similarity in the second dimension, such as the columns. Recently, though, inquiry into co-clustering algorithms, in which both rows and columns in the matrix are clustered simultaneously, has shown that better results can often be obtained [5].

In this progress report, we summarize the our experiments that we have done so far for our project in Section 3. Before that, we will briefly describe the Netflix Prize problem and introduce some related background knowledge in Section 2. Finally in Section 4, we will discuss potential directions for future work.

2 The Netflix Problem and Related Background

In this paper, we consider co-clustering as applied to the Netflix data set. The Netflix dataset is a challenging dataset consisting of over 100 million movie ratings by over 480,000 users [3]. Since collaborative filtering can also be thought of as missing value estimation [2], our goal is to predict the missing values, movies which the user has not rated, from the known ratings of movies from all users. In this paper, we explore co-clustering as a method for estimating these missing values.

Let the data of m users and n movies be represented by an $m \times n$ matrix Z. Each cell $z_{ij} \in [0, 5]$ corresponds to user i's rating of movie j. We seek to partition the m users into k disjoint or hard clusters and the n movies into l disjoint or hard clusters, thus creating a smaller $k \times l$ matrix of co-clusters, which approximates Z.

In [1], the author proposed a Bregman co-clustering algorithm, which can also be considered as a generalized matrix approximation method.

The objective of Bregman co-clustering is to find a partition of m rows and n columns of a data matrix into k row clusters and l column clusters such that the distance between the original matrix and the reconstructed matrix is minimized. The distance measure can be formulated as follows:

$$E(d_\phi(Z, \hat{Z})) = \sum_u^m \sum_v^n w_{uv} d_\phi(z_{uv}, \hat{z}_{uv}) = d_{\phi_w}(Z, \hat{Z})$$

where $\hat{\mathbf{Z}}$ is an approximation of $\mathbf{Z}$ that depends only upon a given co-clustering, w_{uv} is the probability of an entry in $\hat{\mathbf{Z}}$ taking on a particular value z_{uv} and D_ϕ is the Bregman distance with respect to some real-valued convex function ϕ.

As stated in Banerjee's paper, in constructing the approximation matrix $\hat{Z}$ using different co-clustering summary statistics, there are totally 6 different ways (bases). In addition, the I-divergence and squared Euclidean distance instantiated from the general Bregman divergence will lead to a closed form solution for co-clustering.

3 Experiments

In this section, we report the experiments we have done in our course project. First we describe how we pre-process the Netflix data set. Then we discuss our implementation and report some preliminary experimental results.

3.1 Pre-processing

The Netflix datasets come as a collection of multiple files namely the qualifying datasets, the prediction dataset and the probe dataset. The qualifying dataset contains a collection of movies whose ratings by a set of users is withheld. The task is to predict these ratings. The training dataset is a collection of 17770 files, one for each movie in the Netflix system. Each file consists of a Movie ID in the first line followed by a list of tuples of Customer IDs, Ratings (Integral values from 1 to 5) and Dates.CustomerIDs range from 1 to 2649429, with gaps. There are 480189 users totally.

The probe set is similar to the qualifying dataset, it contains a collection of movies each of which has a list of users who have rated them. Unlike the qualifying dataset, the ratings for these movies by the users is available in the training dataset. To test the performance of our algorithm, we can compare the RMSE results obtained by our algorithm on the probe set, against the Netflix's Cinematch system.

It is natural to formulate the ratings between user and movie as a 2-D matrix. We implemented in perl scripts for combining the individual movie files into sparse matrix CCS format. The final data set saved in this format is about 1.5GB. Considering we have only limited storage space for the whole Netflix data, we further convert the data in text format into binary format, which reduced the storage space to less than 500MB.

3.2 Implementation and Experiments

We simultaneously developed the code for Bregman co-clustering algorithm both in C++ and Matlab. Due to the time constraints (considering the the time spent on debugging C++ code) the C++ code is still under progress.

On the other hand, the Matlab code is ready and we tested it on the lhug machine, which took around 2 hours to run on the whole netflix data set. However, at this

Table 1: RMSE on probe set using Bregman co-clustering (Euclidean distance).

Basis	R1/C1	R2/C2	R4/C4	R10/C10	R20/C20	R50/C50
1	1.12962	1.12962	1.12962	1.12962	N/A	N/A
2	1.12962	1.12962	1.12962	0.98376	N/A	N/A
3	1.06879	1.06879	1.06879	1.06879	N/A	N/A
4	1.05282	1.05282	1.05282	1.05282	N/A	N/A
5	0.996456	0.996456	0.99409	0.995524	0.99708	0.97577
6	0.996456	1.05404	1.05902	N/A	N/A	N/A

time, the experiments are still running. We couldn't report complete experimental results here.

We also tested the existing code developed by the author/collaborator in [1]. The preliminary result of RMSE on probe set of Bregman co-clustering using squared Euclidean distance is listed in Table 3.2, which is based on different number of row/column clusters on all bases.

We also tested the part of the existing code using I-divergence. However, we observed that the objective value is increasing instead of decreasing for basis 5 with 20 row/column clusters! We suspect the implementation of the existing code using I-divergence is not correct and will not report any RMSE result here.

4 Discussion and Future Work

We encountered some difficulties in handling the huge Netflix data set on the UTCS machine. Fortunately, we have struggled to save the storage space as much as we can and implemented the runnable MATLAB code for the limited time constraints.

In our future work, we are planning to do the following:

- Completely test our MATLAB code on Netflix Data using both squared Euclidean distance and I-divergence.
- Although we have not observed the occurrence of empty cluster problem in the batch update of our MATLAB code, we would like to explore "local search" or "incremental iterative update" to further improve the performance.
- As one of the major part in our project, we would explore hierarchical co-clustering based on our MATLAB code and will try the Kalman filter on the

resulting tree structure to improve the performance.

References

[1] Arindam Banerjee, Inderjit S. Dhillon, Joydeep Ghosh, Srujana Merugu, and Dharmendra S. Modha. A generalized maximum entropy approach to bregman co-clustering and matrix approximation. *Journal of Machine Learning Research*, 8:1919–1986, August 2007.

[2] Robert M. Bell and Yehuda Koren. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In *ICDM*, 2007.

[3] James Bennet and Stan Lanning. The netflix prize. *KDD Cup and Workshop*, 2007.

[4] H. Cho, I. Dhillon, Y. Guan, and S. Sra. Minimum sum squared residue co-clustering of gene expression data. In *Proceedings of SIAM International Conference on Data Mining 2004*, 2004.

[5] Inderjit S. Dhillon, Subramanyam Mallela, and Dharmedra S. Modha. Information-theoretic co-clustering. In *Proceedings of The Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD-2003)*, 2003.

[6] T. George and S. Merugu. A scalable collaborative filtering framework based on co-clustering. In *Proceedings of IEEE International Conference on Data Mining 2005*, 2005.