Published in final edited form as:
Proc ACM SIGMOD Int Conf Manag Data. 2008 April 25; 24: 1337–1339. doi:10.1109/ICDE.2008.4497548.

A General Framework for Fast Co-clustering on Large Datasets Using Matrix Decomposition

Feng Pan, **Xiang Zhang**, and **Wei Wang**
Department of Computer Science University of North Carolina at Chapel Hill

Abstract

Simultaneously clustering columns and rows (co-clustering) of large data matrix is an important problem with wide applications, such as document mining, microarray analysis, and recommendation systems. Several co-clustering algorithms have been shown effective in discovering hidden clustering structures in the data matrix. For a data matrix of m rows and n columns, the time complexity of these methods is usually in the order of $m \times n$ (if not higher). This limits their applicability to data matrices involving a large number of columns and rows. Moreover, an implicit assumption made by existing co-clustering methods is that the whole data matrix needs to be held in the main memory. In this paper, we propose a general framework, *CRD*, for co-clustering large datasets utilizing recently developed sampling-based matrix decomposition methods. The time complexity of our approach is linear in m and n. And it does not require the whole data matrix be in the main memory. Extensive experimental results on synthetic and several well-known real-life datasets show that *CRD* achieves competitive accuracy to existing co-clustering methods but with much less computational cost.

I. Introduction

Clustering is a fundamental data mining problem with a wide variety of applications. It seeks good partitioning of the data points such that points in the same cluster are similar to each other and the points in different clusters are dissimilar. Many real-life applications involve large data matrices. For example, in text and web log analysis, the term-document data can be represented as contingency table. In biology domain, the gene expression data are organized in matrices with rows representing genes and columns representing experimental conditions. Recently there has been a growing research interest in developing co-clustering algorithms that simultaneously cluster both columns and rows of the data matrix. Co-clustering takes advantage of the duality between rows and columns to effectively deal with the high dimensional data. It has successful applications in gene expression data analysis [1] and text mining [2].

Many formulations of the co-clustering problem have been proposed, such as hierarchical model [3], bi-clustering model [1], pattern-based model [4] and so on. The partitioning-based model, which was first introduced in [3], has attracted much interest, because of the simplicity of the formalization and its close relationships to other well studied problems, such as spectral clustering and matrix decomposition [5], [2], [6], [7], [8], [9]. In this paper, we focus on the partitioning-based co-clustering formulation. Suppose that the data matrix D consists of m rows and n columns. Given input parameters k and l, the partitioning-based co-clustering algorithms try to partition the rows of data matrix into k clusters and columns into l clusters to optimize certain objective functions measuring the quality of the clustering results.

panfeng@cs.unc.edu,xiang@cs.unc.edu,weiwang@cs.unc.edu.

Although theoretically well studied and widely applied, existing co-clustering algorithms usually have the time complexity in the order of $m \times n$. For general data matrices, the information-theoretic co-clustering algorithm introduced in [2] takes $O(t(k + l)mn)$ time to find the clustering results, where t is the number of iterations. Matrix-decomposition (such as nonnegative matrix factorization (NMF) [10]) based co-clustering methods [7], [9], have similar time complexity. In real-life applications, however, the number of rows and columns of the data matrices are usually large. For example, the term-document datasets may contain at least tens of thousands of articles and thousands of words [11]. The high throughput microarray techniques can monitor the expression values of tens of thousands of genes under hundreds to thousands of experimental conditions [12]. Such high time complexity limits the applicability of existing algorithms to these large datasets. Furthermore, these algorithms implicitly make the assumption that the whole data matrix is held in the main memory, since the original data matrix needs to be accessed constantly during the execution of the algorithms.

To address these limitations of existing work, in this paper, we propose a general co-clustering framework, *CRD*[1], for large datasets. This framework is based on recently developed sampling-based matrix decomposition method CUR [13], [14]. Unlike NMF based algorithms, the complexity of *CRD* algorithms is linear in m and n. Moreover, most of the operations in *CRD* involve only the sampled columns and rows. Therefore, we do not require the whole data matrix be in main memory. This is crucial for large datasets. *CRD* can be implemented using different algorithms such as k-means or information-theoretic co-clustering methods. We conduct extensive experiments on both synthetic and several well-known real-life datasets. The experimental results show that *CRD* can be orders of magnitude faster than previous information-theoretic methods and NMF based methods. At the same time, it achieves comparable accuracy to other methods.

II. The CRD Framework

Our *CRD* framework consists of two components.

1. Low rank matrix decomposition: The data matrix is decomposed using a subset of its rows and columns. The decomposition procedure must be fast and accurate.
2. Co-clustering on the subset of rows and columns: The selected rows and columns are co-clustered first. The cluster labels for the rest rows and columns are assigned based on those selected ones. In general, any co-clustering algorithm that optimizes its objective function by alternating the clustering of rows and columns can be used in *CRD*.

Based on the co-clustering results, a subset of the rows/columns may be returned to the decomposition component for re-sampling. The updated decomposition matrices will be sent back to the co-clustering component to be co-clustered again. Figure 1 illustrates the *CRD* framework.

A. Low Rank Row/Column Decomposition

Given a matrix M, $M \in \mathbb{R}^{m \times n}$, let $R = \{\mathbf{r_1}, \mathbf{r_2}, ..., \mathbf{r_m}\}$ represents the set of row vectors of M. We define the low rand row decomposition of M.

Definition 2.1: **Low rank row decomposition**: A low rank *row* decomposition of M approximates the original matrix using a subset of rows.

[1]CRD stands for Co-clustering based on Column and Row Decomposition

$$\tilde{M} = W_R \cdot M_R, \text{where} W_R \in \mathbb{R}^{m \times m'}, M_R \in \mathbb{R}^{m' \times n} \quad (1)$$

where m' is the number of selected rows.

$W_R \cdot M_R$ is a qualified low rank *row* decomposition of M if it satisfies the following constraints,

1. $\|M - \tilde{M}\|_F \leq \varepsilon \|M\|_F$, where $\varepsilon < 1$, is a user-specified approximation rate. $\|\cdot\|_F$ represents the Frobenius matrix norm [15].
2. $M_R = (\mathbf{r}_{u_1} \mathbf{r}_{u_2} \ldots \mathbf{r}_{u_{m'}})^T$, where $\{\mathbf{r}_{u_1} \ldots \mathbf{r}_{u_{m'}}\} \subset R$.
3. Given $M_R = (\mathbf{r}_{u_1} \mathbf{r}_{u_2} \ldots \mathbf{r}_{u_{m'}})^T$, the corresponding rows in W_R have $|w_{u_i i}| > |w_{u_i j}|$, for all $i \in \{1, \ldots, m'\}$ and $j \in \{1, \ldots, m'\} - \{i\}$.

Low rank column decomposition is defined in the similar way. Our decomposition method is based on *CUR* [13]. In general, if the norms of row and column vectors of a data matrix have been calculated, our decomposition method will take $O((m + n)m'n')$ time to find a set of qualified matrices where m' and n' are the numbers of selected rows and columns respectively.

B. Co-clustering Using Row/Column Decomposition

In order to perform co-clustering on large data matrix efficiently, in our *CRD* framework, the co-clustering is performed on the decomposition matrices instead of the original matrix M. It provides significant improvements in both space and time utilization.

One type of co-clustering algorithms cluster the rows and columns of a data matrix alternatively in successive iterations. These algorithms include the Information Theoretic Co-clustering algorithm [2], the Bregman Co-clustering algorithm [5] and the Fully Automatic Cross-association algorithm in [16]. In each iteration, they either keep the row side clusters fixed and re-cluster the columns or keep the column side clusters fixed and re-cluster the rows. It is proved that this single side clustering (row or column) can guarantee the objective function on the co-clustering structure converge to a local minimum (maximum). We call it the **Iterative Single Side Clustering** approach.

In this paper, we use this general approach as the co-clustering component in our *CRD* framework. In each iteration, instead of re-clustering all rows (or columns), we only re-cluster the selected rows or the selected columns in the low rank row/column decomposition matrices. Then we assign cluster label to each row (or column) in M based on the decomposition matrices and the cluster labels of the selected rows (columns). An illustration of co-clustering using the decomposition matrices is shown in Figure 2.

In general, co-clustering using Iterative Single Side Clustering approach can have runtime complexity equal to (or larger than) $O(t(k+l)mn)$ where t is the number of iterations. While our framework only uses $O(t(km'n + ln'm + m'm + n'n))$ plus the time used in matrix decomposition, $O((m + n)m'n')$, which is a one-time cost at the beginning. Since usually we have $m' \ll m$ and $n' \ll n$, our framework is much faster.

III. Experiment

In this section, we present results on the multiple feature dataset to show the efficiency and effectiveness of our *CRD* algorithms.

We implemented two versions of our *CRD* framework using different iterative single side clustering approaches.

- *CRD-ITC*: *CRD* using information-theoretic co-clustering [2].
- *CRD-kmeans*: *CRD* using Euclidian distance (k-means) co-clustering [5].

In order to show the efficiency of our *CRD* framework, we also implemented three recent co-clustering algorithms.

- *ITC*: the original information-theoretic co-clustering algorithm without matrix decomposition[2].
- *Kmeans*: the original Euclidian distance co-clustering algorithm without matrix decomposition[5].
- *ONMF*: the orthogonal nonnegative matrix tri-factorization co-clustering algorithm proposed in [7].

The multiple feature dataset contains 2000 rows and 240 columns. And the 2000 rows are equally divided into 10 classes. We run each algorithm 40 times on the dataset, and plot the distribution of their runtime and row cluster purity in Figure 3. Since the columns do not have class label, the column cluster purity cannot be calculated here.

Each point in Figure 3 represents the runtime or the purity of a single run of one of the algorithms. As we can see, our *CRD* algorithms are about 10 times faster than *ITC* and *Kmeans* and more than 20 times faster than *ONMF*. And if we compare the distribution of cluster purity in Figure 3, we can find that *CRD* algorithms performs as good as the other three algorithms.

IV. Conclusions

In this paper, we proposed a general framework for fast co-clustering on large data, *CRD*. *CRD* has two components. It first decomposes the data matrix into low rank row/column approximation matrices. Then co-clustering algorithms using iterative single-side clustering are used to cluster the approximation matrices. Because of the small size of the approximation matrices, CRD has runtime complexity equal to $O(t(km'n + ln'm + m'm + n'n))$ which is orders of magnitude faster than $O(t(k + l)mn)$, the runtime complexity of the previous co-clustering algorithms. The experiment results show that our framework is both efficient and effective.

References

1. Cheng Y, Church G. Biclustering of experssion data. Proc. of the Eighth Int. Conf. on Intelligent Systems for Molecular Biology. 2000
2. Dhillon IS, Mallela S, Modha DS. Information-theoretic co-clustering. KDD. 2003
3. Hartigan JA. Direct clustering of a data matrix. Journal of the American Statistical Association 1972;67 (337):123–129.
4. Xu X, Lu Y, Tung A, Wang W. Mining shifting-and-scaling co-regulation patterns on gene expression profiles. ICDE. 2006
5. Banerjee A, Dhillon I, Ghosh J, Merugu S, Modha DS. A generalized maximum entropy approach to bregman co-clustering and matrix approximation. KDD. 2004
6. Ding C, He X, Simon H. On the equivalence of nonnegative matrix factorization and. spectral clustering. SDM. 2005
7. Ding C, Li T, Peng W, Park H. Orthogonal nonnegative matrix tri-factorizations for clustering. KDD. 2006
8. Li T. A general model for clustering binary data. KDD. 2005

9. Long B, Zhang Z, Yu P. Co-clustering by block value decomposition. KDD. 2005
10. Lee DD, Seung HS. Learning the parts of objects by non-negative matrix factorization. Nature 1999;401:788–791. [PubMed: 10548103]
11. Slonim N, Tishby N. Document clustering using word clusters via the information bottleneck method. SIGIR. 2000
12. Natsoulis G. Classification of a large microarray data set: Algorithm comparison and analysis of drug signatures. Genome Res 2005;15:724–736. [PubMed: 15867433]
13. Drineas P, Kannan R, Mahoney MW. Fast monte carlo algorithms for matrices iii: Computing a compressed approximate matrix decomposition. SIAM Journal of Computing. 2005
14. Sun J, Xie Y, Zhang H, Faloutsos C. Less is more: Compact matrix decomposition for large sparse graphs. SDM. 2007
15. Golub, G.; Loan, A. Matrix computations. Johns Hopkins University Press; Baltimore, Maryland: 1996.
16. Chakrabarti, D.; Papadimitriou, S.; Modha, D.; Faloutsos, C. Fully automatic cross-associations. ACM SIGKDD'04 Conference Proceedings; 2004;

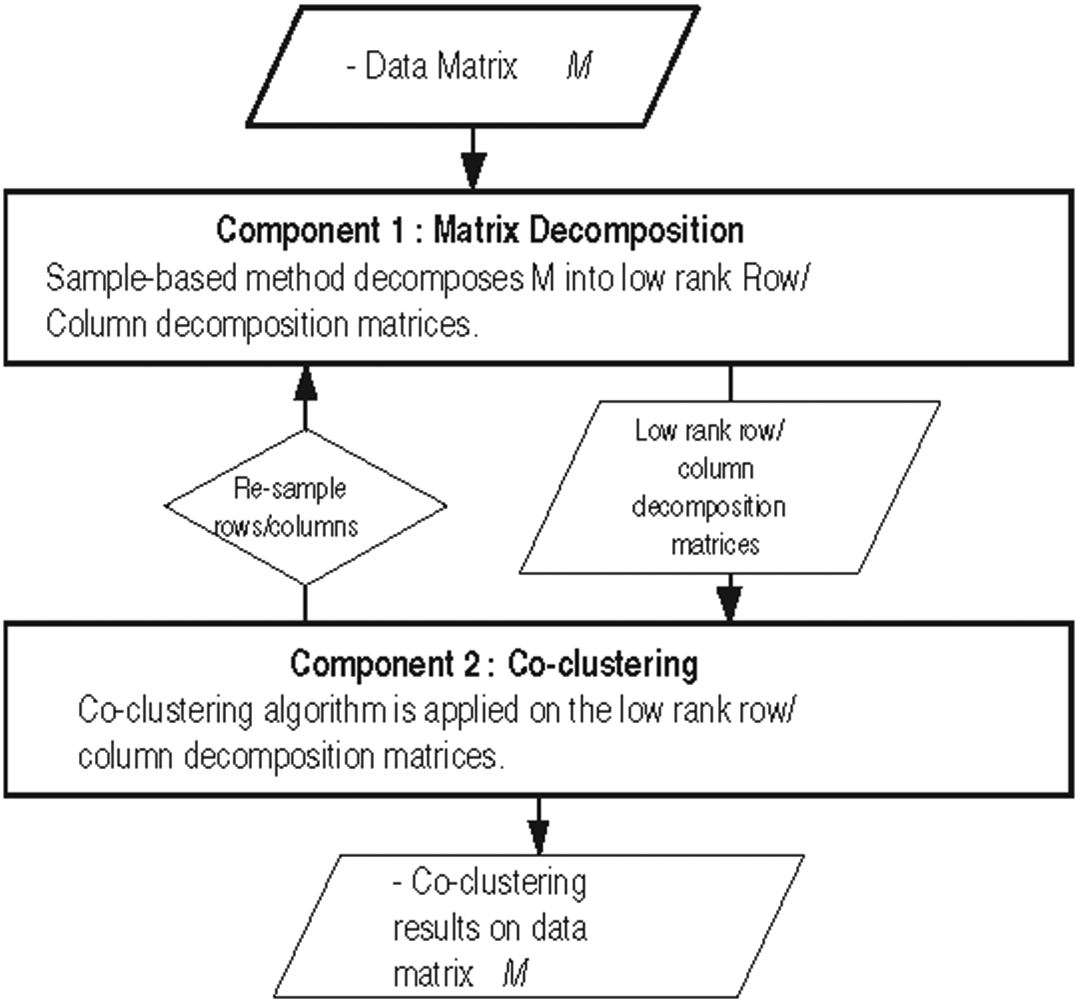

Fig. 1.
CRD framework

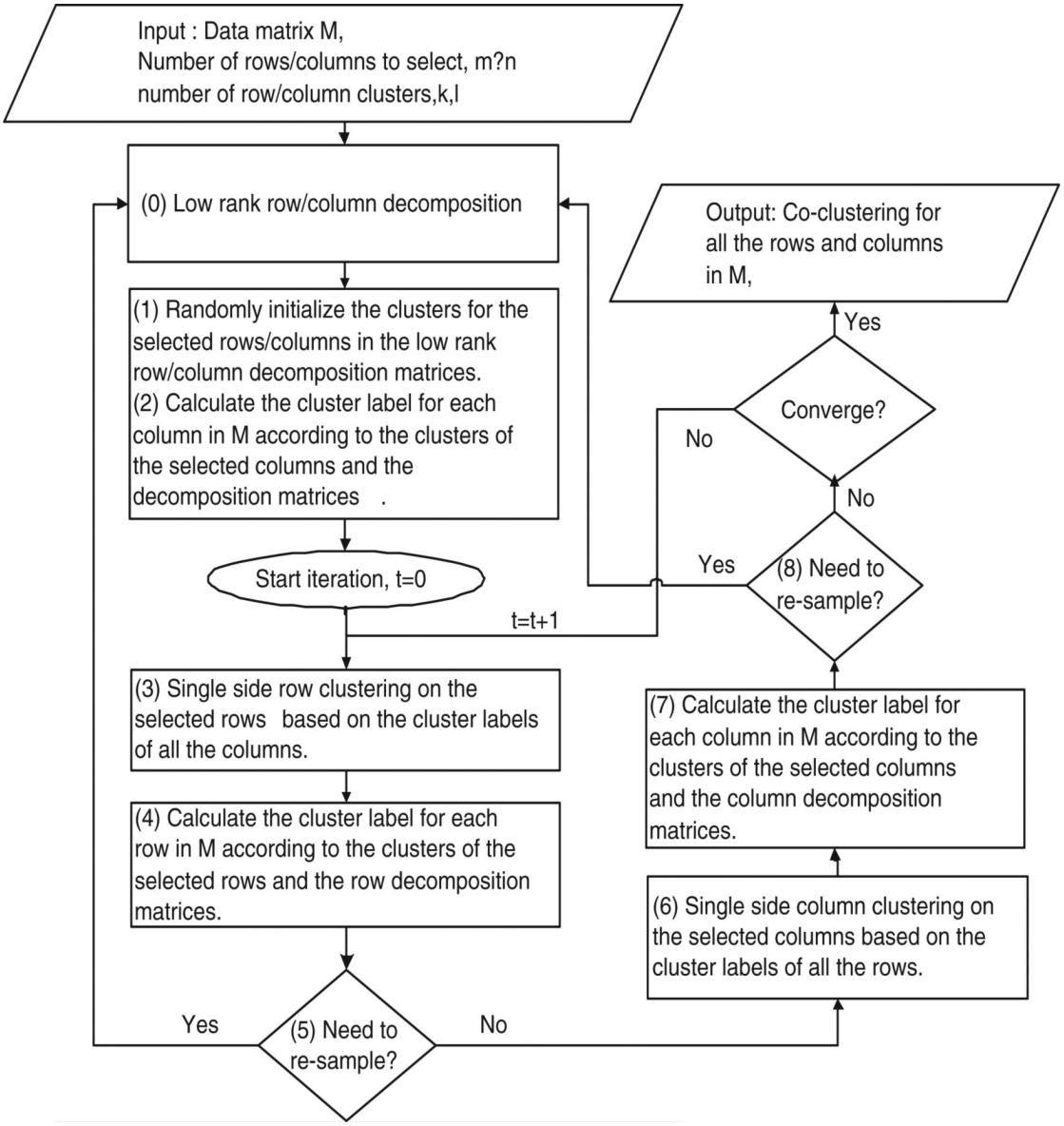

Fig. 2.
Co-clustering using low rank row/column decomposition matrices

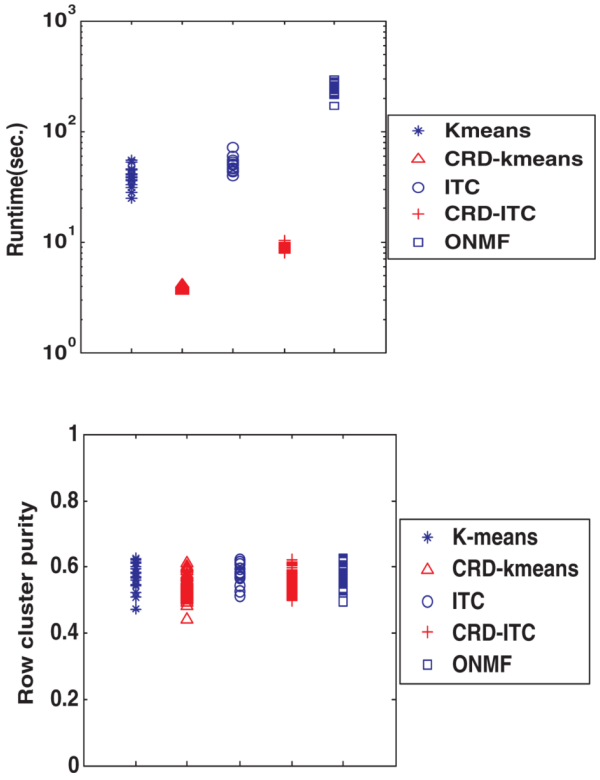

Fig. 3.
Runtime and Row cluster purity performance on Multiple Feature Dataset