Int J Software Informatics, Volume 6, Issue 4 (2012), pp. 523–534
International Journal of Software and Informatics, ISSN 1673-7288

E-mail: ijsi@iscas.ac.cn
http://www.ijsi.org
Tel: +86-10-62661040

Fast Co-Clustering by Ranking and Sampling

Zhao Li and Xindong Wu

(College of Engineering and Mathematical Sciences, University of Vermont, USA)
Email: {zhaoli,xwu}@cems.uvm.edu

Abstract Co-clustering treats a data matrix in a symmetric fashion that a partitioning of rows can induce a partitioning of columns, and vice versa. It has been shown advantageous over tradition clustering. However, the computational complexity of most co-clustering algorithms are costly, and thus limit their effectiveness on large datasets. A recently proposed sampling-based matrix decomposition method can achieve a linear computational complexity, but selected rows and columns can not effectively represent a large sparse dataset, and many unselected rows and columns can not be mapped to the selected rows and columns because they do not share features in common, thus its performance is impaired. To address this problem, we propose a fast co-clustering framework by ranking and sampling that only representative samples are selected for co-clustering, and the remaining samples can be easily labeled by their neighbors in clustered samples. Extensive experiments on large text datasets show that our approach is able to use very few samples to achieve comparable results in linear time compared to state-of-the-art co-clustering algorithms of nonlinear computational complexity.
Key words: co-clustering; ranking; sampling; nonlinear complexity

Li Z, Wu XD. Fast co-clustering by ranking and sampling. *Int J Software Informatics*, **Vol.6, No.4 (2012): 523–534. http://www.ijsi.org/1673-7288/6/i143.htm**

1 Introduction

Clustering analysis is an unsupervised learning that divides similar objects into groups. In a vector space model, where a dataset is formatted as a matrix, most clustering algorithms are conducted in one-way clustering, i.e., each column is an object, data objects are then partitioned based on a similarity measure computed across all the rows. More often than not, simultaneously clustering the rows and columns of a data matrix is required to discover the association between row and column clusters. For example, in recommendation systems, we are interested in finding which group of people is more attracted to a certain group of items, and which group of items is frequently purchased by a certain group of people. By exploiting the duality between rows and columns, co-clustering is fully utilized to deal with sparse and high-dimensional data in many real applications, such as co-clustering the words-documents in text mining[8,7], the genes-experiments in gene expressions[2,6], and the reviewers-movies in recommendation systems[13].

This work is sponsored by US National Science Foundation (NSF) under Grant No.CCF-0905337.
Corresponding author: Zhao Li, Email: zhaoli@cems.uvm.edu
Received 2012-03-15; Accepted 2012-07-30.

Many co-clustering algorithms based on different models have been proposed. A bipartite graph was used in spectral co-clustering to find minimum cut vertex partitions of text data[8]. K-means co-clustering with the objective of minimizing the sum-squared residues was exploited to gene expression data[6]. In Refs. [10,11], authors proposed an informational theoretic co-clustering that aims to maximize mutual information between the clustered entries in a data matrix. Block value decomposition[15], an extension of nonnegative matrix factorization[18,17], was adopted to find the indicator matrices of row and column clusters respectively. Among all these algorithms, the optimization problem of spectral co-clustering is reduced to find a desired partition on one dimensional data that makes it computationally fast, but spectral co-clustering can not deal with different numbers of row and column clusters because the rows and columns are not separated in the process of partitioning. Meanwhile, given an input data matrix consisting of m rows and n columns, and desired numbers of row and column clusters, k and l, respectively, the computational complexity of the rest of algorithms is $O(t(k+l)mn)$, where t is the number of iterations. Thus the overhead of these algorithms is very costly, especially on large datasets where m and n are over tens of thousands. In addition, these methods require the whole data be held in the main memory, which further limits their applications when large memory is hardly available. Recently, a fast co-clustering method based on column and row decomposition (CRD) was proposed to decrease the computational cost[16]. CRD generates a subset of the rows and columns of the data matrix using sampling-based matrix decomposition. Only the selected rows and columns are then co-clustered alternatively, and the rest of rows and columns are labeled by their nearest neighbors during each iteration according to the values obtained in the row and column decomposition. Although this method uses some constraints to ensure the qualification of the sampling-based matrix decomposition and a correct mapping between the selected rows/columns and the unselected rows/columns, it can not provide a good selection of samples. For example, the selected samples come from fewer classes than the desired number of classes. As a result, co-clustering these samples can not produce a correct partition. Also, a large number of unselected samples are unlabeled, i.e, the corresponding values used to map unselected samples to selected ones are zeros.

In this paper, we propose a fast co-clustering framework by ranking and sampling (CRS) to address these problems. The goal of CRS is developing a fast co-clustering algorithm of linear time complexity to achieve comparable results. The key point is how to find those representative samples in order to cover the remaining unselected samples, because we argue that each representative sample is close to the center of its class that the other samples in the class are related to. The CRS framework is conducted by two stages. First, determine the importance of samples based on an appropriate ranking measure, and then select a subset of top-ranked samples. By doing so, the important samples are kept as candidate samples and the problem of finding representative samples is downsized; Second, discriminatively select the representative samples in order to cover as many unselected samples as possible. This is done by combining the ranking of the candidate samples, i.e, iteratively select the sample of the highest ranking value in the current subset of the candidate samples, and remove its neighbors. The samples selected by CRS are more representative that most of the remaining samples can find their labeled neighbors. Experiments show

that CRS can achieve comparable results with very few samples to other algorithms in terms of purity, but with a much less computational cost.

The rest of the paper is organized as follows. First, related work is reviewed. Second, we introduce our proposed algorithm with a detailed analysis. Experiments are conducted on several large datasets to evaluate the effectiveness of our algorithm against other co-clustering algorithms with illustrated results. Finally, a brief conclusion is given in Section 5.

2 Background

In CRD, a fast co-clustering framework utilizing sampling-based matrix decomposition (CUR)[12,20] for large datasets was proposed. Given an input data matrix M, the numbers of rows and columns to select, m' and n', CUR generates three matrices to approximate $M \in \Re^{m\times n}$, i.e.,

$$M \approx B_C \cdot B_U \cdot B_R \tag{2.1}$$

where $B_C \in \Re^{m\times n'}$ and $B_R \in \Re^{m'\times n}$ consist of weighted columns and rows randomly selected from M with a probability in proportional to their norms, respectively, and B_U is computed based on B_C and B_R. CUR is a compressed approximate decomposition of a large matrix, but not designed for clustering rows and columns of the matrix simultaneously. In order to make CUR suitable for the purpose of co-clustering, CRD approximately produces two matrices for both rows and columns, i.e.,

$$\begin{aligned} M &\approx W_R \cdot M_R \\ M &\approx M_C \cdot W_C \end{aligned} \tag{2.2}$$

where $M_R \in \Re^{m'\times n}$ and $M_C \in \Re^{m\times n'}$ are a set of rows and columns without weights respectively; $W_R \in \Re^{m\times m'}$ is a mapping between selected and unselected rows, and $W_C \in \Re^{n'\times n}$ is a mapping between selected and unselected columns. CRD modifies CUR to find a qualified low-rank row and column decomposition based on some constraints to guarantee a good mapping, because CUR produces several different low-rank decompositions for a given data matrix. CRD is conducted using an iterative single side clustering approach[16], which either keeps the row clusters fixed and re-clusters the columns or keeps the column clusters fixed and re-clusters the rows. In CRD framework, K-means and informational theoretic co-clustering algorithms are used. Compared to other co-clustering algorithms, CRD only re-clusters the selected rows and columns, and maps the unselected rows and columns based on W_R and W_C, respectively. The computational complexity of the CRD algorithm is claimed to be $O(t(km'n + ln'm + m'm + n'n) + (m+n)m'n')$. However, CRD inherits the property of CUR that sampled rows and columns are selected with a probability, thus they are not necessarily from all different classes of the data. Moreover, because CRD can not effectively deal with large sparse data, there are several rows and columns of all zero entries in W_R and W_C, respectively. In other words, many rows and columns can not find their labeled neighbors. However, in the iterative single side clustering approach the labels of all rows and columns are required.

3 Co-clustering by Ranking and Sampling (CRS)

Given a dataset of several classes, each class should have at least one representative or centered sample that other samples are associated with. It is difficult to find all of those representative samples. However, we can determine the importance of each sample, and then discriminatively select those candidate samples that possibly represent different classes for the purpose of co-clustering. Another benefit of taking advantage of determining the importance of each sample is that we can remove less important samples to reduce the size of data for the sampling process. Thus, we introduce a fast co-clustering algorithm by ranking and sampling for this regard.

3.1 Ranking and sampling

In fact, those representatives samples can be obtained by ranking their importance among all samples. For example, in graph theory the adjacency matrix of a set of nodes defines the linking structure on which several algorithms are applied to get the ranking of all nodes. Two popular ranking algorithms, PageRank[1,5] and hypertext induced topics search (HITS)[14,3], are theoretically well studied in the literature of link analysis. General speaking, a link from node A to node B denotes a type of endorsement, for instance, considering B an authority on a subject. Link analysis is used to assign an authority value for each node in a connected direct graph. With the intuition that good authorities are usually pointed by good authorities, Pagerank is a probability distribution that a random selected web page follows a random outgoing link with a probability of α, and jumps to a random web page with a probability of $1-\alpha$. It has been utilized in social network analysis with a huge success. On the other hand, by using hubs and authorities to define a recursive relationship between web pages, HITS exploits the idea, with a slight difference, that good hubs point to good authorities and good authorities are pointed by good hubs. The HITS algorithm is in fact a power-method eigenvector computation. Given a data matrix $M \in \Re^{m\times n}$, we can compute the values for authority nodes x and the values for hub nodes y as follows:

$$\begin{aligned} x &= M^T y \\ y &= Mx. \end{aligned} \tag{3.3}$$

Substituting these two equations we get

$$\begin{aligned} x &= M^T M x \\ y &= M M^T y. \end{aligned} \tag{3.4}$$

x and y can be solved by finding eigenvectors for $M^T M$ and MM^T, for example, given $M = USV^T$,

$$\begin{aligned} M^T M &= VS^T U^T USV^T = V(S^T S)V^T \\ MM^T &= USV^T VS^T U^T = U(SS^T)U^T \end{aligned} \tag{3.5}$$

where $S^T S$ is a diagonal matrix with the eigenvalues. Thus, x and y are the first vectors of left and right matrices V and U, respectively, i.e., the first eigenvectors of $M^T M$ and MM^T.

To rank both rows and columns of M, PageRank usually needs a computational complexity of $O(m^2+n^2)$[4] while that of HITS is $m^2n+O(m)+n^2m+O(n)$, because, according to Equation 3.5, M^TM and MM^T only need to be calculated once, and $O(m)$ or $O(n)$ for per eigenvector[19]. However, it is noted that if we assume that rows only take hub values, and columns only take authority values, x and y can be directly obtained from the first left and right eigenvectors, U_1 and V_1, since $M = USV^T$. Thus, the computational complexity is reduced to $O(m+n)$. It is worthy noting that U_1 and V_1 may consist of mixed values (negative and nonnegative). To get correct rankings of rows and columns, we need to get the absolute values of U_1 and V_1. After obtaining the rankings of rows and columns, we can obtain these samples, each of which has the ranking value above the average ranking value of the samples, as the candidate samples. By doing so, most important samples are kept, and the size of samples to select is significantly reduced.

Provided with the rankings of both rows and columns of a data matrix, to get a certain number of representative samples , we can choose the top-ranked samples in the candidate set and remove some of their nearest neighbors since the samples are more likely from the same class of their nearest neighbors. It is noted that sorting the ranking of all samples at the beginning is not necessary since some nearest neighbors will be deleted. Thus our algorithm can be conducted by iteratively choosing the sample of the highest ranking value in the current candidate set and deleting a fixed number of its nearest neighbors until a certain number of samples are selected. Ideally, if the data is evenly distributed, all representative samples in each class can be selected. Our ranking and sampling algorithm for rows/columns is described in Algorithm 1.

Algorithm 1 Ranking and sampling for co-clustering

Input: An input matrix $M \in \Re^{m\times n}$, m' rows and n' columns to select.
Output: Selected lists of rows and columns, R and C.
Method:

Calculate the first left and right eigenvectors, $|U_1|$ and $|V_1|$, as the row and column rankings, respectively, and the average ranking value of $|U_1|$ and $|V_1|$.
Put all row or column samples in L, and delete the sample in L whose ranking value is lower than the average ranking value.
while $|R| \leqslant m'$ or $|C| \leqslant n'$ **do**
  Choose the sample s of the highest ranking value in L, and add it to $\{R\}$ or $\{C\}$.
  Compute the distance vector between s and $\{L-s\}$ based on Euclidean distance.
  Delete $\lfloor(|L|/m')\rfloor$ or $\lfloor(|L|/n')\rfloor$ nearest neighbors of s from $\{L\}$.
end while

3.2 A framework of co-clustering by ranking and sampling

A framework of co-clustering by ranking and sampling (CRS) is illustrated in Fig. 1. Assuming the size of candidate row/column samples in L is reduced to m''/n'' after removing the less important samples. Then the size of the candidate samples is gradually downsized by m''/m' or n''/n'. The computational complexity of the ranking and sampling algorithm is $mn+(m'+1)m''n/2+(n'+1)n''/2+O(m+n+m''m'+n''n')$, in which mn is used to calculate the norms of all row and column

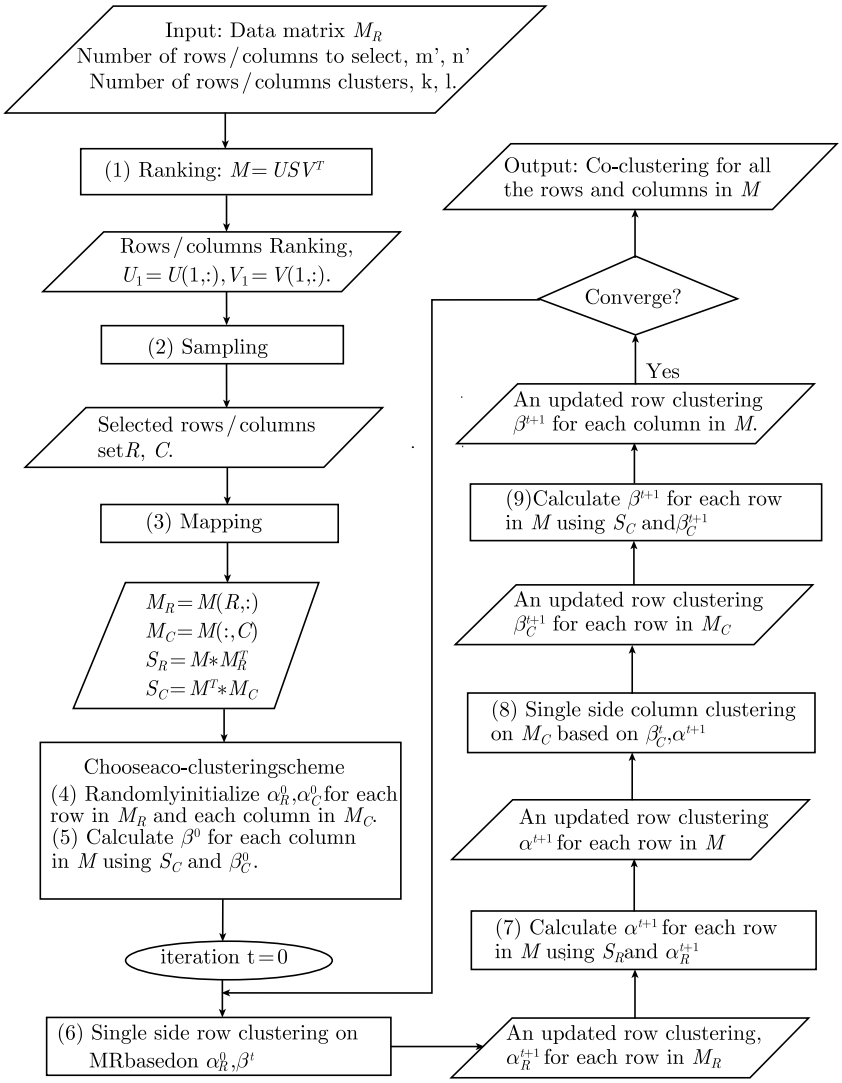

Figure 1. A framework of co-clustering by ranking and sampling

vectors; $(m'+1)m''n/2 + (n'+1)n''/2$ is for computing the row and column distance vectors; $O(m+n)$ accounts for computing the first left and right eigenvectors; $O(m+n+m''m'+n''n')$ is for selecting representative samples and removing their nearest neighbors. In our design, we adopt the same iterative single side clustering approach as used in Ref. [16]. The computational complexity of iterative single side clustering approach is $O(t(km'n + ln'm))$. After co-clustering sampled rows/-columns, row/column cluster labels are mapped to all rows/columns, we get the correlation between unselected and selected samples by calculating their similarity matrices $S_R = M * M(R,:)^T$ and $S_C = M^T * M(:,C)$. This is done once by using a runtime of $m'm + n'n$. This simple mapping can not guarantee no unlabeled samples, however, the more samples being chosen, the less unlabeled samples. We will show that in our method it is useful even selecting very few samples. CRD considers re-sampling which is infrequently happened as reported in Ref. [16] to get a better selection when neces-

sary. In this paper, we do not consider re-sampling. The computational complexity of CRD without re-sampling is $mn+m'm+n'n+O((m+n)m'n')+O(t(km'n+ln'm))$ while that of CRS is $mn + m'm + n'n + (m' + 1)m''n/2 + (n' + 1)n''/2 + O(m + n + m''m' + n''n') + O(t(km'n + ln'm))$, where m'' is the size of the initial candidate samples, each of which has ranking value higher than the average ranking value of all samples. It is much smaller than the original size of all samples. In practice, we could use different criteria to get a smaller m'' in order to further decrease the computational cost. Generally, m' and n', the numbers of samples to select, are very small and can be considered as constants. Therefore, the CRS framework achieves an execution linear time since $m' \ll m''$, and $n' \ll n''$.

4 Experiments

4.1 Datasets

In this section, we conduct our experiments on 6 large text datasets from the CLUTO toolkit[1] to evaluate the performance of CRS with other co-clustering algorithms in terms of running time and purity. They were obtained from a variety of sources. The *hitech* and *reviews* datasets were derived from the San Jose Mercury newspaper articles that were used in the TREC collection[2]. The *La12* and *Sports* datasets were extracted from the articles of the Los Angeles Times contained in TREC collections. the *Ng3sim* dataset was 3 overlapping groups about politics (talk.politics.misc, talk.politics.guns and talk.politics.mideast) extracted from the 20 Newsgroups[3]; the *K1b* dataset was obtained from the WebACE project, a subset of news articles from Reuters[4]. All these databases associated with the toolkit were already preprocessed[21]. Moreover, those words that appear in two or fewer documents were further removed, and each data matrix was normalized. This toolkit provides a good representation of different characteristics which are summarized in Table 1.

Table 1 Description of the document datasets

Dataset	#Words	#Documents	#Classes
Hitech	10080	2301	6
K1b	11715	2340	6
Ng3sim	15810	2998	3
Reviews	18483	4069	5
La12	20009	6279	6
Sports	14870	8580	7

4.2 Measurements

In our experiments, each document dataset is represented by a term-document matrix. Since we do not have class labels for rows (terms), the evaluation of clustering quality is conducted on columns (documents). We use purity and runtime to evaluate the quality of all algorithms in this paper. Purity gives the average ratio of

[1] `http://glaros.dtc.umn.edu/gkhome/cluto/cluto/download`
[2] `http://trec.nist.gov/data.html`
[3] `http://people.csail.mit.edu/jrennie/20Newsgroups/`
[4] `http://trec.nist.gov/data/reuters/reuters.html`

a dominating class in each cluster according to the cluster size and is defined as:

$$Purity = \frac{1}{N}\sum_{m}\max_{n}(|k_m \cap c_n|), \tag{4.6}$$

where N is the size of samples, $\{k_1, k_2, \cdots, k_n\}$ is the set of clusters, and $\{c_1, c_2, \cdots, c_m\}$ is the set of classes. Higher values of the purity indicate better clustering.

4.3 Performance evaluation

We also use the information theoretical co-clustering (ITC) and k-means co-clustering (KCC) algorithms for iterative single side clustering[16]. Therefore, 7 algorithms are considered: CRD-KCC, CRD-KCC, CRS-ITC, CRS-KCC, ITC, KCC, and BVD. For sampling based methods, CRD and CRS, as discussed before, it is possible to have unselected samples unlabeled. For those unlabeled samples, they can be randomly labeled for simplicity. All algorithms were implemented in Matlab R2010b on a 4 dual-processor, dual-core (Opteron 2220) IBM x3455s, with 6GB of ram. All results were gathered by 50 independent trails. Meanwhile, no more than 200 iterations was allowed when they failed to converge at a stopping criterion.

First, we evaluate the CRD and CRS algorithms with different numbers of samples on all datasets. We varied the number of selected rows/columns from 10 to 50. The purity comparison with error-bar is shown in Fig. 2, and the runtime comparison is illustrated in Fig. 3. Generally, CRS-KCC and CRS-ITC outperform CRD-KCC and CRD-ITC by average improvement of 26.4% and 48.6%, respectively, on all datasets. Increasing the number of selected rows/columns can also improve the purity of the CRS-ITC algorithm. However, it affects the CRS-KCC algorithm. The

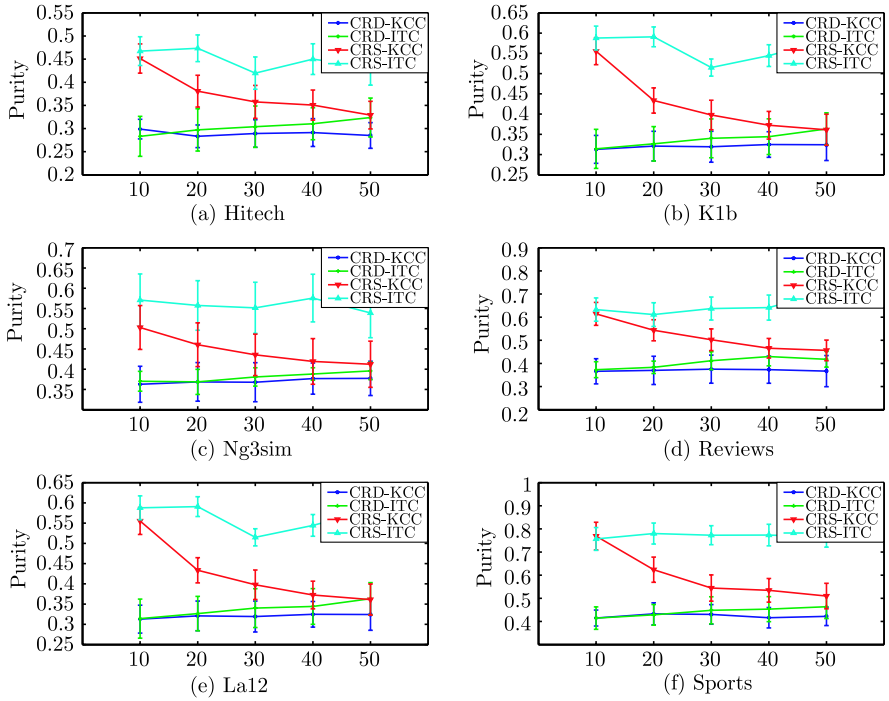

Figure 2. Purity comparison with different numbers of samples on all datasets

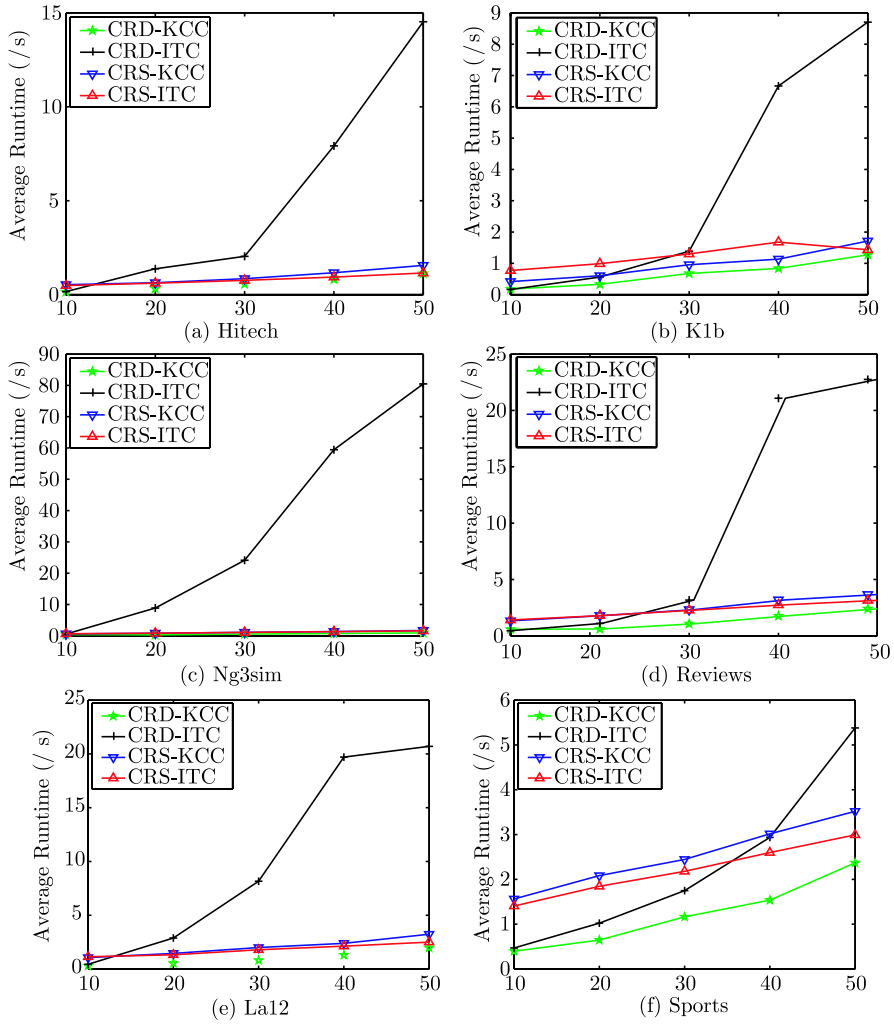

Figure 3. Average runtime comparison with different numbers of samples on all datasets

reason is that when a sample is selected a certain number of its nearest neighbors are deleted based on Euclidean distance. The number of the neighbors to delete is decided by the number of samples to select. Therefore, the more sample selected, the less neighbors deleted, thus less discriminative the selected samples are in terms of Euclidean distance used in the KCC algorithm. As for runtime comparison, compared to CRD-ITC, the runtime of CRS-KCC and CRS-ITC are slightly increased, and very close to CRD-ITC. However, the runtime of CRD-KCC grows rapidly when increasing the number of samples. As discussed before, the size of candidate samples, m'', could affect runtime. In fact, on average, approximately 25%/45% rows/columns are selected as candidate samples.

To compare all algorithms, we need to choose the number of rows/columns for sampling based algorithms. Since the maximum number of classes among all datasets is 7 on the *Sports* dataset. For the CRD algorithms we use the default setting as reported in Ref. [16]: 20 rows/columns are selected, and for our CRS algorithms, 10 rows/columns are sufficient based on the performance illustrated above. The purity

and runtime comparisons are illustrated in Fig. 4 and Fig. 5, respectively. For runtime comparison, because the co-clustering algorithms without performing sampling use significant runtime we take the common logarithm with base 10 in order to reduce wide-ranging quantities to smaller scopes. As shown in the Fig. 4, CRS outperforms CRD in both KCC and ITC algorithms on all datasets. It is also worthy noting that, on datasets *Hitech* and *Sports*, CRS-KCC and CRS-ITC outperform the KCC, ITC, and BVD algorithms, and on other datasets the results are very comparable. However, as reported in Fig. 5, the runtime of the CRS is orders of magnitude faster than other co-clustering algorithms without performing sampling. In our experiments the runtime of CRS on all datasets is no more than 5 seconds. Both CRD and CRS achieve the computational complexity of linear time, however CRS prevails over CRD because CRS obtains a better selection of representative samples, and in contrasted to CRD most unselected samples are well mapped to selected samples. Since in each dataset the number of rows is way more than the number of columns, In our experiments we found that CRD had approximately 20% column samples and 60% row samples not mapped. However, in our method CRS, there is almost no unlabeled column samples samples(only 3 unlabeled column occurred on the dataset *Ngs3sim*) and very few row samples unlabeled. Therefore we argue that ranking and discriminative sampling are very efficient in finding representative samples and mapping unselected samples, thus lead to a significant improvement of clustering performance for sampling based algorithms.

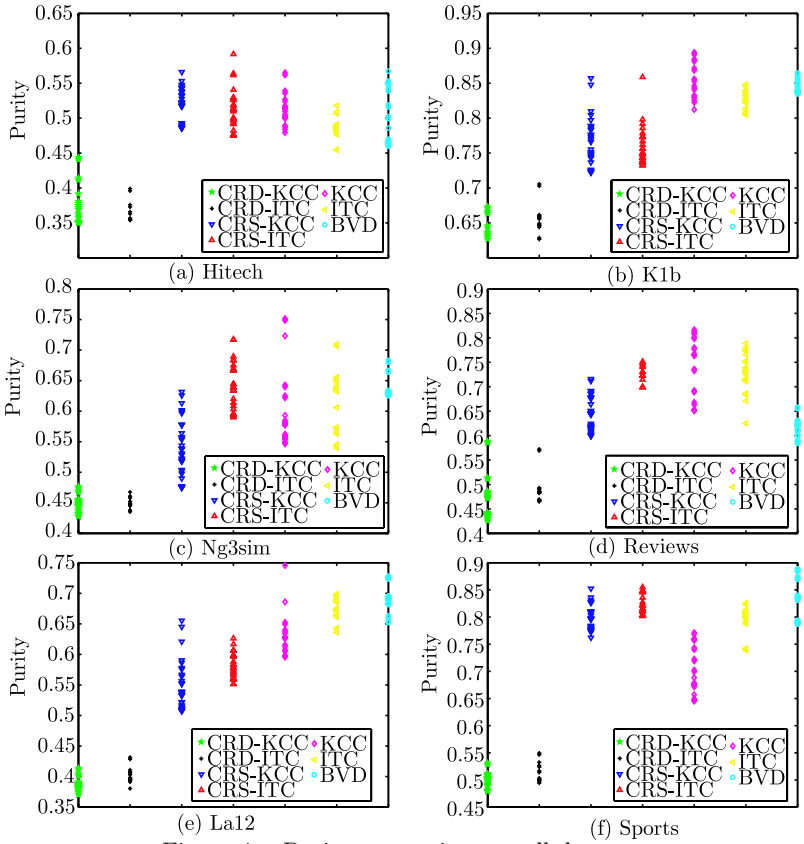

Figure 4. Purity comparison on all datasets

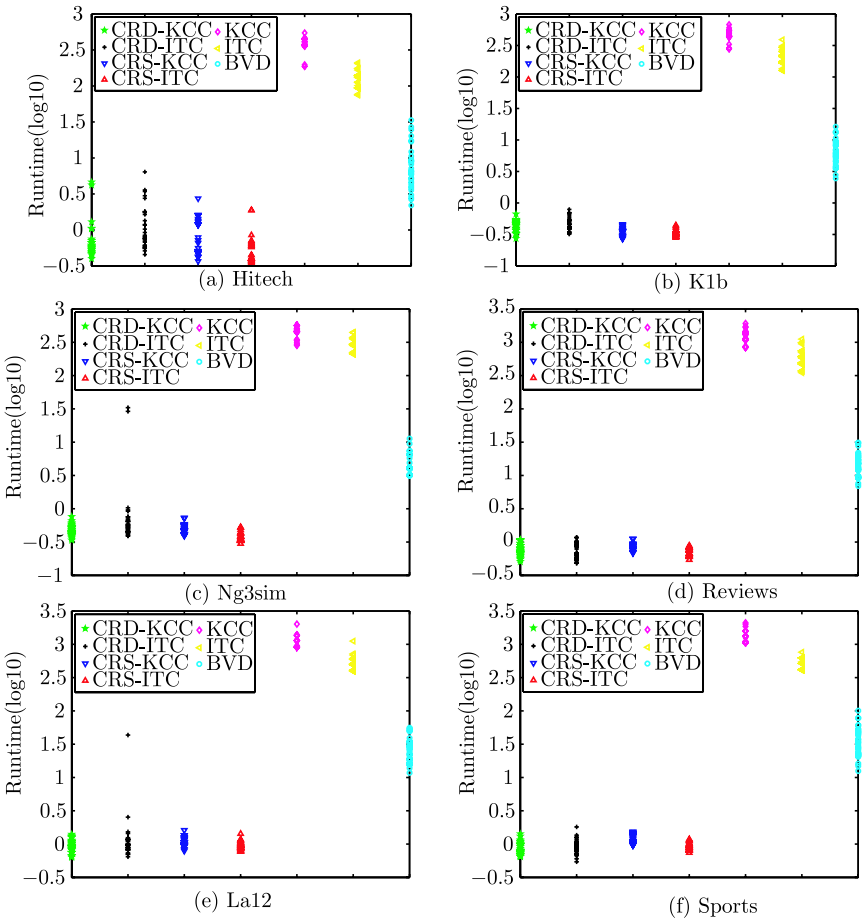

Figure 5. Runtime(Log10) comparison on all datasets

5 Conclusions

In this paper, we have introduced a fast co-clustering approach based on ranking and sampling (CRS) with the aim of developing a fast and efficient co-clustering algorithm. Ranking is used for determining the importance of samples, and sampling is for discriminatively selecting representative samples for the purpose of co-clustering. Using the idea of the HITS algorithm, we could assign each row of a data matrix a hub value and each column an authority value to rank both columns and rows, which can be done in linear time by computing the first left and right eigenvectors. Based on the rankings of rows and columns, more important samples are selected as candidate samples, on which the sampling strategy is conducted by iteratively choosing the sample of the highest ranking value from the candidate samples and removing its nearest neighbors until a certain number of samples are chosen. This discriminative sampling efficiently obtains those representative samples that most unselected samples can find labeled neighbors from the selected samples. The CRS algorithms also achieve a linear runtime, but more importantly, CRS outperforms CRD even using fewer samples than CRD. Meanwhile, CRS is very comparable to other co-clustering algorithms without performing the sampling strategy that are very

costly in time and space.

References

[1] Brin S, Page L. The anatomy of a large-scale hypertextual web search engine. Proc. of the seventh international conference on World Wide Web. WWW. 1998. 107–117.

[2] Cheng Y, M G. Church: Biclustering of expression data. Proc. of the 8th International Conference on Intelligent Systems for Molecular Biology. 2000. 93–103.

[3] Chakrabarti S, Dom BE, Gibson D, Kleinberg J, Kumar R, Raghavan P, Rajagopalan S, Tomkins A. Mining the link structure of the World Wide Web. IEEE Computer, 1999, 32: 60–67.

[4] Haveliwala T. Efficient Computation of PageRank. [Technical Report]. Stanford, 31, 1999.

[5] Page L, Brin S, Motwani R, Winograd T. The PageRank Citation Ranking: Bringing Order to the Web. [Technical Report]. Stanford, 66, 1999.

[6] Cho H, Dhillon IS, Guan YQ, Sra S. Minimum sum-squared residue co-clustering of gene expression data. Proc. of the 4th SIAM International Conference on Data Mining. 2004. 114–125.

[7] Ding C, Li T, Peng W, Park H. Orthogonal nonnegative matrix tri-factorizations for clustering. Proc. of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Philadelphia, PA, USA. 2006. 126–135.

[9] Dhillon IS. Co-clustering documents and words using bipartite spectral graph partitioning. Proc. of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2001. 269–274.

[10] Kluger Y, Basri R, C JT, Gerstein M. Spectral biclustering of microarray data: coclustering genes and conditions. Genome Res, April 2003, 13: 703–716.

[11] Dhillon IS, Mallela S, Modha DS. Information-theoretic co-clustering. Proc. of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2003. 89–98.

[12] Banerjee A, Dhillon I, Ghosh J, Merugu S, Modha DS. A generalized maximum entropy approach to bregman co-clustering and matrix approximation. Journal of Machine Learning Research, December 2007, 8: 1919–1986.

[13] Drineas P, Kannan R, Mahoney MW. Fast monte carlo algorithms for matrices ii: Computing a low-rank approximation to a matrix. SIAM J. Comput., July 2006, 36: 158–183.

[14] George T, Merugu S. A scalable collaborative filtering framework based on co-clustering. Proc. of the Fifth IEEE International Conference on Data Mining. ICDM '05. Washington, DC, USA. 2005. 625–628.

[15] Kleinberg JM. Hubs, authorities, and communities. ACM Comput. Surv., December 1999, 31.

[16] Long B, Zhang ZF, Yu PS. Co-clustering by block value decomposition. Proc. of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2005. 635–640.

[17] Pan F, Zhang X, Wang W. Fast co-clustering on large datasets utilizing sampling-based matrix decomposition. Proc. of the 2008 ACM SIGMOD International Conference on Management of Data. SIGMOD '08. ACM. 2008. 173–184.

[18] Lee DD, Seung HS. Algorithms for non-negative matrix factorization. Proc. of 15th Annual Conference on Neural Information Processing Systems. 556-562. 2001.

[19] Lee DD, Seung HS. Learning the parts of objects by non-negative matrix factorization. Nature, October 1999, 401: 788–791.

[20] Stanley D, Stanley K. The performance of finding eigenvalues and eigenvectors of dense symmetric matrices on distributed memory computers. Proc. of the Seventh SIAM Conference on Parallel Proceesing for Scientific Computing. SIAM. 1994. 528–533.

[21] Sun JM, Xie YL, Zhang H, Faloutsos C. Less is more: Sparse graph mining with compact matrix decomposition. Stat. Anal. Data Min., February 2008, 1:6–22.

[22] Zhong S, Ghosh J. Generative model-based document clustering: a comparative study. Knowledge Information System, 2005, 8(3): 374–384.