Proc. 16th Italian Symp. on Advanced Database Systems SEBD 2008,
Mondello (PA), Italy, June 22-25, 2008. pp. 279-286.

Numerical Data Co-clustering via Sum-Squared Residue Minimization and User-defined Constraint Satisfaction[⋆]

Ruggero G. Pensa[1] and Jean-François Boulicaut[2]

[1] Pisa KDD Laboratory, ISTI-CNR, I-56124 Pisa, Italy
ruggero.pensa@isti.cnr.it
[2] INSA-Lyon, LIRIS CNRS UMR5205, F-69621 Villeurbanne cedex, France
jean-francois.boulicaut@insa-lyon.fr

EXTENDED ABSTRACT

Abstract. Co-clustering aims at computing a bi-partition that is a collection of co-clusters: each co-cluster is a group of objects associated to a group of attributes and these associations can support interpretations. We consider constrained co-clustering not only for extended must-link and cannot-link constraints (i.e., both objects and attributes can be involved), but also for interval constraints that enforce properties of co-clusters when considering ordered domains. We propose an iterative co-clustering algorithm which exploits user-defined constraints while minimizing the sum-squared residues, i.e., an objective function introduced for gene expression data clustering by Cho et al. (2004).

1 Introduction

In many application domains, the data analyst has to consider possibly large data sets that record numerical values of given properties for given objects (say $objects \times features$ matrices). Exploratory data analysis processes often make use of clustering techniques to get insights about global patterns within the data, i.e., to propose partitions of objects and/or of properties such that a grouping quality measure is optimized. Many clustering methods can compute partitions but suffer from the lack of explicit cluster characterization. This has motivated the research on conceptual clustering, e.g., the co-clustering approaches described in [6, 2]. The objective of co-clustering is to compute co-clusters that are associations of (possibly overlapping) sets of objects with sets of properties. A co-clustering algorithm computes simultaneously linked partitions on both rows and columns. Co-clustering has been well studied in the context of gene expression data analysis because it provides valuable information about putative regulation mechanisms and biological functions [5]. Intuitively, a co-cluster extracted from a gene expression data set (a matrix that records gene expression

[⋆] This paper summarizes the results presented in [8].

values for given genes in given biological samples) denotes a set of genes with similar expression profiles along its associated set of biological samples.

We are designing co-clustering methods that compute more relevant bi-partitions thanks to the satisfaction of user-defined constraints. Given a co-clustering algorithm, the analyst has generally a weak control on the clusters he/she obtains. Typically, he/she can decide for ad-hoc parameter settings which are quite operational and conceptually far from the declarative specification of desired properties. A co-clustering algorithm tries to optimize an objective function (e.g., the sum of squared residues [6]) but it may also ensure that some constraints that express his/her domain knowledge are satisfied (e.g., the fact that some objects and/or properties have to be together or not). Enforcing constraints can lead to lower values for the objective functions, and it is clear that combining objective function optimization and the satisfaction of other user-defined constraints is challenging. The last 5 years, several researchers have studied "single-sided" constrained clustering for simple types of primitive constraints, mainly the so-called must-link and cannot-link constraints (see [3] for an up-to-date survey). To the best of our knowledge, constrained co-clustering has been rarely studied. In [9], we have studied co-cluster discovery when at least one of the dimensions is ordered and when interval constraints are defined w.r.t. orders. A typical application concerns kinetic gene expression data analysis. In this case, objects denote gene expression level measurements performed for successive time points. For an organism like Plasmodium Falciparum [4], we see that during its life cycle, groups of genes are activated and then inhibited, being somehow characteristic of development stages. Using interval constraint can support the discovery of such groups from experimental data. However, the method presented in [9] is dedicated to Boolean data analysis and it is based on a two-step strategy that first extracts local patterns and then post-process them to compute the co-clusters. Furthermore, the method from [9] cannot guarantee the satisfaction of all the constraints in the computed bi-partition. In this paper, we introduce a new constraint-based co-clustering approach which builds a bi-partition directly from numerical matrices. The resulting bi-partition satisfies the user-defined constraints while the algorithm has been optimizing the objective function proposed in [6], namely the sum-squared residues.

2 A constrained co-clustering setting

Let $X \in \mathbb{R}^{m \times n}$ denote a data matrix to be mined. In this paper, we will talk about rows and columns instead of objects and properties. Let x_{ij} be the element corresponding to row i and column j. For instance, x_{ij} might contain the expression level of gene i in the experimental condition j. Let $x_{i.}$ and $y_{.j}$ denote the vectors associated to, respectively, row i and column j. A co-clustering $C^{k \times l}$ over X produces simultaneously a set of $k \times l$ co-clusters (a partition C^r into k groups of rows associated to a partition C^c into l groups of columns). To obtain a first quality criterion, we first try to optimize a certain objective function. Let us assume an objective function $f(X, C^{k \times l})$, an optimization constraint

$c_{opt}(f, X, C^{k \times l})$ is satisfied iff $C^{k \times l} = \text{argmin}_{\phi \in \mathcal{L}_{\mathcal{C}^{k \times l}}} f(X, \phi)$ where $\mathcal{L}_{\mathcal{C}^{k \times l}}$ is the the collection of all possible co-clusterings. One can be interested in other kinds of constraints which are now defined.

Definition 1 (must-link/cannot-link). *If rows i_a and i_b (resp. columns j_a and j_b) are involved in a* ***must-link*** *constraint, denoted $c_=(i_a, i_b)$ (resp. $c_=(j_a, j_b)$), they must be in the same cluster of $C^r = r_1, \ldots, r_k$ (resp $C^c = c_1, \ldots, c_k$). If rows i_a, i_b (resp. columns j_a and j_b) are involved in a* ***cannot-link*** *constraint, denoted $c_{\neq}(i_a, i_b)$ (resp. $c_{\neq}(j_a, j_b)$), they cannot be in the same cluster of $C^r = r_1, \ldots, r_k$ (resp $C^c = c_1, \ldots, c_k$).*

Let us now assume that all the elements j (resp. i) are ordered, i.e., $\forall j_a, j_b$ s.t. $j_a \leq j_b$, $j_a \prec j_b$ (the same property holds for rows). In microarray data, the order might be related to the sampling time of each DNA chip (say experiment) j. Then, It could also be interesting to search for co-clusters which are coherent with such an order. For this purpose, we can enforce an interval constraint.

Definition 2 (interval constraint). *If an order ($\preceq$) is defined over the column set (resp. row set), an* ***interval*** *constraint over this set, denoted $c_{int}(C^c)$, specifies that each cluster in C^c has to be an interval: $\forall c \in C^c$, if $j_a, j_b \in c$ then $\forall j_c$ such that $j_a \preceq j_c \preceq j_b$, $j_c \in c$.*

We want a co-clustering algorithm which is able to take into account such constraints while trying to optimize the retained objective function.

3 Using sum-squared residues

Our approach to constrained co-clustering is based on an iterative algorithm that minimizes the sum of squared residues. This objective function has been introduced in [6] for unconstrained co-clustering applied to gene expression data. It is an adaptation of a measure designed for local pattern discovery [5].

Given a data matrix $X \in \mathbb{R}^{m \times n}$, we search for a partition of X into k row clusters, and l column clusters. Let I be the set of indices of the rows belonging to a row cluster, and J the set of indices of the columns belonging to a column cluster. The sub-matrix of X determined by I and J is called a *co-cluster*. We use the definition of residue from [5].

Definition 3 (residue). *Given an element x_{ij} of X, the residue of x_{ij} in the co-cluster defined by the sets of indices I and J, and whose respective cardinalities are $|I|$ and $|J|$, is given by*

$$h_{ij} = x_{ij} - x_{Ij} - x_{iJ} + x_{IJ} \tag{1}$$

where $x_{IJ} = \frac{\sum_{i \in I, j \in J} x_{ij}}{|I| \cdot |J|}$, $x_{Ij} = \frac{\sum_{i \in I} x_{ij}}{|I|}$ and $x_{iJ} = \frac{\sum_{j \in J} x_{ij}}{|J|}$.

Algorithm 1: ConsCoClust(X,k,l,$\mathcal{M}_r$,$\mathcal{M}_c$,$\mathcal{C}_r$,$\mathcal{C}_c$)

```
Input: Data matrix $X$, $k$, $l$, cannot-link sets $\mathcal{C}_r$ and $\mathcal{C}_c$, collections $\mathcal{M}_r$ and $\mathcal{M}_c$
Output: Matrices $R$ and $C$
Initialize $R$ and $C$;
$\Delta = 1$; $\tau = 10^{-2}||X||^2$; $t = 0$; $obj^t = ||(I - RR^T)X(I - CC^T)||^2$;
while $\Delta > \tau$ do
    $t = t + 1$; $X^C = (I - RR^T)XC$; $X^P = (I - RR^T)X$;
    foreach $1 \leq j \leq n$ do
        $L = \emptyset$;
        if $\exists M_v \in \mathcal{M}_c$ $s.t.$ $j \in M_v$ then
            foreach $j_v \in M_v$ do
                $L = L \cup \{1 \leq c \leq l \mid \nexists j_c \mid \gamma^t[j_c] = c \wedge c_{\neq}(j_v, j_c) \in \mathcal{C}_c\}$;
            end
            $\gamma^t[M_v] = \text{argmin}_{c \in L} \frac{\sum_{j_v \in M_v} ||X^P_{.j} - n_c^{-1/2} X^C_{.c}||^2}{|M_v|}$;
        else
            $L = \{1 \leq c \leq l \mid \nexists j_c \mid \gamma^t[j_c] = c \wedge c_{\neq}(j, j_c) \in \mathcal{C}_c\}$;
            $\gamma^t[j] = \text{argmin}_{c \in L} ||X^P_{.j} - n_c^{-1/2} X^C_{.c}||^2$;
        end
    end
    Update $C$ using $\gamma$;
    $X^R = R^T X(I - CC^T)$; $X^P = X(I - CC^T)$;
    {Reassign rows};
    $obj^t = ||(I - RR^T)X(I - CC^T)||^2$; $\Delta = |obj^t - obj^{t-1}|$;
end
```

Let $H = [h_{ij}] \in \mathbb{R}^{m \times n}$ denote the matrix of residues computed using the previous definition. The objective function to be minimized is the sum of squared residues [6] computed as follows:

$$||H||^2 = \sum_{I,J} ||h_{IJ}||^2 = \sum_{I,J} \sum_{i \in I, j \in J} h_{ij}^2 \tag{2}$$

We can rewrite the residue matrix in a more compact form. Let us introduce the matrices $R \in \mathbb{R}^{m \times k}$ and $C \in \mathbb{R}^{n \times l}$ which are defined as follows: each element (i, r) $(1 \leq r \leq k)$ of R is equal to $m_r^{-1/2}$ if i is in co-cluster r (m_r is the number of rows in r), 0 otherwise. Each element (j, c) $(1 \leq c \leq l)$ of the matrix C is equal to $n_c^{-1/2}$ if j is in c (n_c being the number of columns in c), 0 otherwise. The residue matrix becomes:

$$H = (I - RR^T)X(I - CC^T) \tag{3}$$

Our approach uses the so-called "ping-pong" technique to process alternatively (applying a k-means method) columns and rows. Thus, matrix C is updated only after determining the nearest column cluster for each column (and similarly for rows). For that, we can decompose the objective function in terms

Algorithm 2: IntCoClust(X,k,l)

```
Input: Data matrix $X$, $k$ and $l$
Output: Matrices $R$ and $C$
Initialize $R$, $C$, $left$, $right$; $(\star)$
$\Delta = 1$; $\tau = 10^{-2}||X||^2$; $t = 0$; $obj^t = ||(I - RR^T)X(I - CC^T)||^2$;
while $\Delta > \tau$ do
    $t = t + 1$; $X^C = (I - RR^T)XC$; $X^P = (I - RR^T)X$;
    foreach $1 \leq c \leq l$ do
        $stop = false$;
        while $c > 1 \wedge stop = false \wedge right[c] > left[c]$ do
            $j = left[c]$;
            if $||X^P_{.j} - n_{c-1}^{-1/2} X^C_{.c-1}||^2 < ||X^P_{.j} - n_c^{-1/2} X^C_{.c}||^2$ then
                $\gamma^t[j] = c - 1$; $left[c] = left[c] + 1$; $right[c-1] = right[c-1] + 1$;
            else
                $stop = true$;
            end
        end
        {Process right-side frontiers};
    end
    Update $C$ using $\gamma$;
    $X^R = R^T X(I - CC^T)$; $X^P = X(I - CC^T)$;
    {Reassign rows}$(\star\star)$;
    $obj^t = ||(I - RR^T)X(I - CC^T)||^2$; $\Delta = |obj^t - obj^{t-1}|$;
end
```

of columns. Given $X^P = (I - RR^T)X$, $X^C = (I - RR^T)XC$, and $\hat{X}^P = (I - RR^T)XCC^T = X^C C^T$, we can rewrite the objective function as follows:

$$||X^P - \hat{X}^P||^2 = \sum_{c=1}^{l} \sum_{j \in J_c} ||X^P_{.j} - \hat{X}^P_{.j}||^2 = \sum_{c=1}^{l} \sum_{j \in J_c} ||X^P_{.j} - n_c^{1/2} X^C_{.c}||^2$$

We have a similar result for the rows of the matrix. Then, matrices X^C and X^R correspond to the cluster centroids for columns and rows respectively.

We introduce now our algorithms to solve the constrained co-clustering problem. More details about the algorithms, as well as the complexity analysis, can be found in [8].

The transitivity of the must-link constraint is a well known property. We can then transform a set of must-link constraints over rows into a collection $\mathcal{M}_r = M_1, \ldots, M_N$, where each M_i is a set of rows involved by the same transitive closure of must-link constraints. Let us denote $\mathcal{M}_c$ the same set built for columns and let $\mathcal{C}_r$ and $\mathcal{C}_c$ be the sets of cannot-link constraints for rows and columns respectively.

Algorithm 1 enables to co-cluster data when conjunctions of must-link and cannot-link constraints are given. It starts with some initialization (e.g., random initialization) of matrices C and R. During each iteration, the algorithm associates each column (resp. row) to the nearest column (resp. row) cluster which does not introduce any cannot-link violation. If a column (resp. row) is involved in a must-link constraint, the algorithm associates the whole set of columns (resp. rows) involved in the transitive closure of this constraint to the column (resp. row) cluster such that the average distance is minimum, and controlling that there is no cannot-link constraint which is violated by this operation. Then the algorithm updates the matrix C (resp. R) following the assignment schema resulting from the previously described operations. This process is iterated until the diminution of the objective value turns to be smaller than a user-defined threshold τ.

Algorithm 2 enables to solve the satisfaction problem for the interval constraint (the part concerning row assignment ($\star\star$) is omitted here). The initialization ($\star$) of partitions interested by this constraint should produce a number l (resp. k) of intervals over columns (resp. rows). Then, the assignment process only considers the frontiers between intervals. More precisely, it first processes the left frontier, then the right frontier iteratively. A column (resp. row) can be assigned to the adjacent interval if the distance is smaller than the distance computed over its original interval. In this case, we continue processing the remaining columns (resp. rows). When the left frontier and the right frontier of an interval correspond to the same column (resp. row), the algorithm starts to process the next frontier. If there is no necessity to reassign the column (resp. row), the algorithm stops the current frontier processing and it skips to the following one. Notice that, contrary to [9], the satisfaction of the interval constraint on the computed bi-partition is here ensured.

4 Experimental validation

We have studied the impact of our constraint-based co-clustering approach on two well-documented microarray data sets, plasmodium [4] and drosophila [1]. The first one concerns the transcriptome of the intraerythrocytic developmental cycle of Plasmodium Falciparum, i.e., a causative agent of human malaria. The data provide the expression profile of 3 719 genes in 46 biological samples. Each sample corresponds to a time point of the developmental cycle: it begins with merozoite invasion of the red blood cells, and it is divided into three main phases, the ring, trophozoite and schizont stages. The second data set concerns the gene expression of the Drosophila melanogaster during its life cycle [1]. The expression levels of 3 944 genes are evaluated for 57 sequential time periods divided into embryonic, larval and pupal stages.

Results for must-link et cannot-link constraints We measured the impact of combining constraints over row sets and column sets on plasmodium data. For this purpose, we selected a bi-partition among the unconstrained co-clustering results. In particular, we chose the co-clustering results with the minimum objective function value obtained at the end of the iterative process. This value is about 1.99×10^4. Then, we generated 20 random sets of constraints involving gene and biological conditions. The number of genes involved in these constraints is about 5% to 10% of the total size of the gene set. For biological conditions, this number is about 15% to 25%. As the initial partitions were randomly generated, our algorithm has been executed 20 times for each group of constraints. To evaluate the agreement between the selected bi-partition and the ones discovered by the constrained algorithm, we used the adjusted Rand index [7]. Results obtained by using constraints have been compared with those obtained by the unconstrained algorithm (see Table 1). We can see that using constraints gives

	AR_r	AR_c	$\|\|H\|\|^2$	Number of Iterations
Constrained	**0.88**	**0.73**	$\mathbf{2.16 \times 10^4}$	**9.18**
Unconstrained	0.70	0.43	2.21×10^4	9.35

Table 1. Adjusted Rand index, final objective function value and number of iterations.

rise to an obvious improvement of the agreement between the two bi-partitions (AR_r and AR_c are the agreement indexes measured on rows and columns respectively). As side effects, the average number of iterations needed for co-clustering is slightly smaller. Moreover, specifying constraints enables to improve the final value of the objective function. Indeed, the $||H||^2$ value decreases by about 2%. Notice that, since the sets of constraints are randomly selected, these experiments include results achieved through constraints which can either positively influence the results or do not introduce any quality improvement. However, as average clustering results are good, we can conclude that our approach enables to obtain more relevant bi-partitions according to prior biological knowledge.

Results for the interval constraint We evaluated the added value of interval constraint satisfaction on drosophila data. Here, our goal is to rediscover the three phases of the drosophila life cycle (i.e., we set the number of clusters ($k = l = 3$) and this is the only information we use). We compared the adjusted Rand index for the constrained and unconstrained versions of our algorithm and for a collection of 20 randomly initialized runs. The results (see Table 2), show that using an interval constraint enables to find more accurately the three stages of the drosophila life cycle (the measured improvement for the adjusted Rand index is about 85%). Notice that the final value of the objective function for the unconstrained version of the algorithm is better than for the constrained version. It means that the structure which our algorithm is able to discover is unlikely to be the global optimum for this data set. Despite of this, the unconstrained algorithm has never managed to find intervals.

5 Conclusion

Improving co-cluster relevancy remains a difficult task in real-life exploratory data analysis processes. First, it is hard to capture subjective interestingness

	AR	$\|\|H\|\|^2$	Number of Iterations
Constrained	**0.76**	8.23×10^4	11.60
Unconstrained	0.41	7.73×10^4	14.60

Table 2. Adjusted Rand index, final objective function value and number of iterations.

aspects, e.g., the analyst's expectation given her/his domain knowledge. Next, when these expectations can be declaratively specified, using them during bi-partition computation is challenging. We have proposed a new constrained co-clustering algorithm. We explained how to exploit user-defined constraints like must-link, cannot-link, and interval constraints when co-clustering numerical matrices. Applications on kinetic gene expression data analysis have been considered. A short-term perspective is to combine properly the strategies for exploiting must-link and cannot-link constraints one one hand, and interval constraints on another hand. We may also study the possibility to discover overlapping co-clusters.

Acknowledgements. This research is partially funded by EU contracts IQ IST-FP6-516169 and GeoPKDD IST-FP6-014915.

References

1. M.N. Arbeitman, E.E. Furlong, F. Imam, E. Johnson, B.H. Null, B.S. Baker, M.A. Krasnow, M.P. Scott, R.W. Davis, and K.P. White. Gene expression during the life cycle of drosophila melanogaster. *Science*, 297:2270–2275, september 2002.
2. A. Banerjee, I. Dhillon, J. Ghosh, S. Merugu, and D.S. Modha. A generalized maximum entropy approach to bregman co-clustering and matrix approximation. *Journal of Machine Learning Research*, 8:1919–1986, 2007.
3. S. Basu, I. Davidson, and K. Wagstaff (Editors). *Constrained Clustering: Advances in Algorithms, Theory and Applications.* Chapman & Hall/CRC Press, Data Mining and Knowledge Discovery Series, 2008. In press.
4. Z. Bozdech, M. Llinás, B. Lee Pulliam, E.D. Wong, J. Zhu, and J.L. DeRisi. The transcriptome of the intraerythrocytic developmental cycle of plasmodium falciparum. *PLoS Biology*, 1(1):1–16, October 2003.
5. Y. Cheng and G. M. Church. Biclustering of expression data. In *Proceedings ISMB 2000*, pages 93–103, San Diego, USA, 2000. AAAI Press.
6. H. Cho, I. S. Dhillon, Y. Guan, and S. Sra. Minimum sum-squared residue co-clustering of gene expression data. In *Proceedings SIAM SDM 2004*, Lake Buena Vista, USA, 2004.
7. L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, 1985.
8. R. G. Pensa and J-F. Boulicaut. Constrained co-clustering of gene expression data. In *Proceedings SIAM SDM 2008*, pages 25–36, Atlanta, GA, USA, 2008.
9. R. G. Pensa, C. Robardet, and J-F. Boulicaut. Towards constrained co-clustering in ordered 0/1 data sets. In *Proceedings ISMIS 2006*, volume 4203 of *LNCS*, pages 425–434, Bari, Italy, 2006. Springer.