Bicluster enumeration using Formal Concept Analysis

Víctor Codocedo and Amedeo Napoli

LORIA - CNRS - INRIA - Université de Lorraine, BP 239, 54506 Vandœuvre-les-Nancy.
victor.codocedo@loria.fr, amedeo.napoli@loria.fr,

Abstract. In this work we introduce a novel technique to enumerate constant row/column value biclusters using formal concept analysis. To achieve this, a numerical data-table (standard input for biclustering algorithms) is modelled as a many-valued context where rows represent objects and columns represent attributes. Using equivalence relations defined for each single column, we are able to translate the bicluster mining problem in terms of the partition pattern structure framework. We show how biclustering can benefit from the FCA framework through its robust theoretical description and efficient algorithms. Finally, we show how this technique is able to find high quality biclusters (in terms of the mean squared error) more efficiently than a state-of-the-art bicluster algorithm.

1 Introduction

Biclustering has become a fundamental tool for bioinformatics and gene expression analysis [4]. Different from standard clustering where objects are compared and grouped together based on their full descriptions, biclustering generates groups of objects based on a subset of their attributes, values or conditions. Thus biclusters are able to represent object relations in a local scale instead of the global representation given by an object cluster [12]. In this sense, biclustering has many elements in common with Formal Concept Analysis (FCA) [6]. In FCA objects are grouped together by the attributes they share in what is called a formal concept. Furthermore, formal concepts are arranged in a hierarchical and overlapping structure denominated a concept lattice. Hence a formal concept can be considered as a bicluster of objects and attributes representing relations in a local scale, while the lattice structure gives a description in the global scale. FCA is not only analogous to biclustering, but has much to offer in terms of mining techniques and algorithms [10]. The concept lattice can also provide biclusters with an overlapping hierarchy which has been reported as an important feature for bicluster analysis [15]. Recently, some approaches considering the use of FCA algorithms to mine biclusters from a numerical data-table have been introduced showing good potential [8, 7]. In this work, we present a novel technique for lattice-based biclustering using the pattern structure framework [5], an extension of FCA to deal with complex data. More specifically, we propose a technique for mining biclusters with similar row/column values, a specialization of biclustering focused on mining attributes with coherent variations, i.e. the difference between two attributes is the same for a group of objects [12]. We show that, by the use of partition pattern structures [1], we can find high quality maximal biclusters (w.r.t. the mean squared error). Finally, we compare our approach with a standard constant row value algorithm [3], showing the capabilities and limitations of our approach.

The remainder of this paper is organized as follows. The basics of biclustering are introduced in Section 2. Section 3 presents our approach and Section 4 presents the experiments and initial findings of our biclustering technique. Finally, Section 5 concludes our article and presents some new perspectives of research.

2 Biclustering definitions

A numerical data-table is a matrix $\mathcal{M}$ where $\mathcal{M}_{ij}$ indicates the value of an object $\mathtt{g}_i \in \mathtt{G}$ w.r.t. the attribute $\mathtt{m}_j \in \mathtt{M}$ with $i \in [1..|\mathtt{G}|]$ and $j \in [1..|\mathtt{M}|]$ ($|\cdot|$ represents set cardinality). A bicluster of $\mathcal{M}$ is a submatrix $\mathcal{B}$ where each value $\mathcal{B}_{ij}$ satisfies a given restriction. According to [4, 12], there are five different restrictions which we summarize in Table 1.

Constant values	$\mathcal{B}_{ij} = c$	Within the submatrix, all values are equal to a constant $c \in \mathbb{R}$ ($\mathbb{R}$ indicates real values).
Constant row values	$\mathcal{B}_{ij}{=}c + \alpha_i$	Within the submatrix, all the values in a given row i are equal to a constant c and a row adjustment $\alpha_i \in \mathbb{R}$.
Constant column values	$\mathcal{B}_{ij}{=}c + \alpha_j$	Within the submatrix, all the values in a given column j are equal to a constant c and a column adjustment $\alpha_j \in \mathbb{R}$.
Coherent values	$\mathcal{B}_{ij}{=}c + \alpha_i + \beta_j$	Within the submatrix, all the values in a given column j are equal to a constant c, a row adjustment α_i and a column adjustment β_j. Instead of addition, the model can also consider multiplicative factors.
Coherent evolution		Values in the submatrix induce a linear order.

Table 1: Types of biclusters.

Similar values instead of constant values When noise is present in a data-table, it is difficult to search for constant values. Several approaches have tackled this issue in different ways, e.g. by the use of evaluation functions [14], equivalence relations [2, 13] and tolerance relations [7]. The most common way is establishing a threshold $\theta \in \mathbb{R}$ to enable the similarity comparison of two different values $w_1, w_2 \in \mathbb{R}$. We say that $w_1 \simeq_\theta w_2$ (values are similar) iff $|w_1 - w_2| \leq \theta$. Thus, constant values are a special case of similar values when $\theta = 0$. Using this, we can redefine the first three types of biclusters as follows:

1. Similar values: $\mathcal{B}_{ij} \simeq_\theta \mathcal{B}_{kl}$.

2. Similar row/column values:
   (a) Similar row values: $\mathcal{B}_{ij} \simeq_\theta \mathcal{B}_{il}$.
   (b) Similar column values: $\mathcal{B}_{ij} \simeq_\theta \mathcal{B}_{kj}$.

Example 1. *With $\theta = 1$, Table 2 shows in its upper left corner a bicluster with similar values (dark grey). The upper right corner represents a similar column bicluster (light grey). Lower left corner considering $\{\mathtt{g}_3, \mathtt{g}_4\}$ and $\{\mathtt{m}_1, \mathtt{m}_2\}$ (not marked in the table) represents a similar row bicluster.*

	$\mathtt{m}_1$	$\mathtt{m}_2$	$\mathtt{m}_3$	$\mathtt{m}_4$	$\mathtt{m}_5$
$\mathtt{g}_1$	1	2	2	1	6
$\mathtt{g}_2$	2	1	1	0	6
$\mathtt{g}_3$	2	2	1	7	6
$\mathtt{g}_4$	8	9	2	6	7

Table 2: Bicluster with similar values ($\theta = 1$).

	$\mathtt{m}_1$	$\mathtt{m}_3$
$\mathtt{g}_2$	2	1
$\mathtt{g}_3$	2	1

Table 3: Constant column bicluster.

3 Biclustering using partition pattern structures

The pattern structure framework is an extension of FCA proposed to deal with complex data [5]. Partition pattern structures are an instance of the pattern structure framework proposed to mine functional dependencies among attributes of a database [1] dealing with set partitions. In the following, we provide the specifics of partition pattern structures where the main definitions are given in [5].

Let $\mathtt{G}$ be a set of objects, $\mathtt{M}$ a set of attributes and $\mathcal{M}$ a data-table of numerical values where $\mathcal{M}_{ij}$ contains the value of attribute (column) $\mathtt{m}_j \in \mathtt{M}$ in object (row) $\mathtt{g}_i \in \mathtt{G}$. A partition $\mathtt{d} = \{p_i\}$ of the set $\mathtt{G}$ can be formalized as a collection of components p_i such as:

$$\bigcup_{p_i \in \mathtt{d}} p_i = \mathtt{G} \qquad p_i \cap p_j = \emptyset \; ; (p_i, p_j \in \mathtt{d}, i \neq j)$$

Two partitions can be ordered by the *coarser-finer* relation where we say that a partition $\mathtt{d}_1 = \{p_i\}$ is a refinement of $\mathtt{d}_2 = \{p_j\}$ (or $\mathtt{d}_2$ is a coarsening of $\mathtt{d}_1$) iff $\forall \; p_i \in \mathtt{d}_1, \exists \; p_j \in \mathtt{d}_2, \; p_i \subseteq p_j$. We denote this as $\mathtt{d}_1 \sqsubseteq \mathtt{d}_2$ where $\mathtt{d}_1, \mathtt{d}_2 \in \mathtt{D}$ is the space of all partitions of the set $\mathtt{G}$.

Let us define the mapping function $\delta : \mathtt{M} \to \mathtt{D}$, which assigns to each attribute in $\mathtt{M}$ the partition it generates over the set of objects $\mathtt{G}$, as follows:

$$\delta(\mathtt{m}_j) = \{[\mathtt{g}_i]_{m_j} \mid \mathtt{g}_i \in \mathtt{G}\} \tag{1}$$

$$[\mathtt{g}_i]_{\mathtt{m}_j} = \{\mathtt{g}_k \in \mathtt{G} \mid \mathcal{M}_{ij} = \mathcal{M}_{kj}\} \tag{2}$$

Where $[\mathtt{g}_i]_{\mathtt{m}_j}$ is the equivalence class of $\mathtt{g}_i$ w.r.t. attribute $\mathtt{m}_j$, i.e. the set of rows in data-table $\mathcal{M}$ which have the same value in column $\mathtt{m}_j$ as row $\mathtt{g}_i$. Since the set of equivalence classes for a given attribute generates a partition over $\mathtt{G}$, it comes naturally that $\delta(\mathtt{m}_j) \in \mathtt{D}$ for any $\mathtt{m}_j \in \mathtt{M}$.

It is easy to show that the order in the space of object partitions $\mathtt{D}$ defines a complete lattice for which the similarity operator $\sqcap$ for any two partitions $\mathtt{d}_1, \mathtt{d}_2 \in \mathtt{D}$ is defined as follows:

$$\mathtt{d}_1 \sqcap \mathtt{d}_2 = \bigcup p_i \cap p_j \tag{3}$$

$$\mathtt{d}_1 \sqsubseteq \mathtt{d}_2 \iff \mathtt{d}_1 \sqcap \mathtt{d}_2 = \mathtt{d}_1 \tag{4}$$

Then, a *partition pattern structure* is determined by the triple $(\mathtt{M}, (\mathtt{D}, \sqcap), \delta)$ in which the following derivation operators for $\mathtt{B} \subseteq \mathtt{M}$ and $\mathtt{d} \in \mathtt{D}$ are defined:

$$\mathtt{B}^{\square} = \bigsqcap_{\mathtt{m} \in \mathtt{B}} \delta(\mathtt{m}) \tag{5}$$

$$\mathtt{d}^{\square} = \{\mathtt{m} \in \mathtt{M} \mid \mathtt{d} \sqsubseteq \delta(\mathtt{m})\} \tag{6}$$

Similarly to standard FCA, we have that $(\mathtt{B}, \mathtt{d})$ is a partition pattern concept (pp-concept) when $\mathtt{B}^{\square} = \mathtt{d}$ and $\mathtt{d}^{\square} = \mathtt{B}$ and that for two pp-concepts $(\mathtt{B}_1, \mathtt{d}_1)$ and $(\mathtt{B}_2, \mathtt{d}_2)$, the order between them is given by $(\mathtt{B}_1, \mathtt{d}_1) \leq (\mathtt{B}_2, \mathtt{d}_2) \iff (\mathtt{B}_1 \subseteq \mathtt{B}_2)$ or $(\mathtt{d}_2 \sqsubseteq \mathtt{d}_1)$. Pp-concepts determines biclusters as pairs $(p, \mathtt{B})$ where p is a component of the partition pattern $\mathtt{d}$. It should be noticed that to keep consistency with previous notation, we write biclusters as pairs $(p, \mathtt{B})$ (p represent rows and $\mathtt{B}$ represent columns), while pp-concepts are written inversely $(\mathtt{B}, \mathtt{d})$ ($\mathtt{B}$ is the extent and $\mathtt{d}$ is the intent of $(\mathtt{B}, \mathtt{d})$).

Proposition 1. *Let* $(\mathtt{B}, \mathtt{d})$ *be a pp-concept, then for any partition component* $p \in \mathtt{d}$ *each pair* $(p, \mathtt{B})$ *corresponds to a constant column value bicluster.*

The proof of this proposition is straightforward considering that each pair $(p, \mathtt{B})$ represents a submatrix the columns of which were selected using an equivalence relation, i.e. the values in the columns are the same.

We say that a bicluster $(p, \mathtt{B})$ is maximal iff adding an object to p or an attribute to $\mathtt{B}$ does not result in a bicluster, i.e. $(p \cup \{\mathtt{g}\}, \mathtt{B})$ and $(p, \mathtt{B} \cup \{\mathtt{m}\})$ are not biclusters. While pp-concepts are maximal (closed under $(\cdot)^{\square}$), biclusters corresponding to pairs $(p, \mathtt{B})$ are not always maximal. This is due to the fact that pp-concepts are maximal w.r.t. the partitions and not w.r.t. the individual components of those partitions. Nevertheless, maximal biclusters are still easy to identify.

Proposition 2. *Let* $(\mathtt{B}_1, \mathtt{d}_1), (\mathtt{B}_2, \mathtt{d}_2)$ *be two pp-concepts such as* $(\mathtt{B}_1, \mathtt{d}_1) \leq (\mathtt{B}_2, \mathtt{d}_2)$*. Let* $p \subseteq \mathtt{G}$ *be a component of a partition. If* $p \in \mathtt{d}_1$ *and* $p \notin \mathtt{d}_2$ *then the bicluster corresponding to* $(p, \mathtt{B}_1)$ *is maximal.*

Proof. Given definitions in Equations 2, 5 and 6, we have that for $(\mathtt{B}_1, \mathtt{d}_1)$ and for any $\mathtt{g}_i \in p$, the following is true:

$$p = \bigcap_{\mathtt{m}_j \in \mathtt{B}_1} \{\mathtt{g}_k \in \mathtt{G} \mid \mathcal{M}_{ij} = \mathcal{M}_{kj}\} \tag{7}$$

Consequently, for any other object $\mathtt{g}_h \in \mathtt{G}$, such as $\mathtt{g}_h \notin p$, we have $\mathcal{M}_{ij} \neq \mathcal{M}_{hj}$. Hence, the pair $(p + \{\mathtt{g}_h\}, \mathtt{B})$ cannot be a bicluster.

Let $\mathtt{B}_2 = \mathtt{B}_1 + \{\mathtt{m}_j\}$ for any $\mathtt{m}_j \in \mathtt{M}$, we show that $(p, \mathtt{B}_2)$ cannot be a cluster by contradiction. Let $(p, \mathtt{B}_2)$ be a bicluster. Then, there exists the pp-concept $(\mathtt{B}_2, \mathtt{B}_2^{\square})$ such as $p \in \mathtt{B}_2^{\square}$. If it does, then it is necessarily a direct super concept of $(\mathtt{B}_1, \mathtt{d}_1)$. However, this contradicts the definition $p \notin \mathtt{B}_2^{\square}$. □

Supporting similar values: In general, it is not possible to support similar value biclusters as described in Section 2 using the partition pattern structures framework. This is due to the fact that the restriction $\mathcal{B}_{ij} \simeq_\theta \mathcal{B}_{kl} \iff |\mathcal{B}_{ij} - \mathcal{B}_{kl}| \leq \theta$ is not transitive and hence, it is not an equivalence but a tolerance relation [10] which do not necessarily generates partitions over the set of objects. However, the setting to support this scenario is only slightly different from the partition pattern structures framework. We do not provide its description for the sake of simplicity.

Nevertheless, through the use of interval of values we can get a close representation of similar value biclusters considering that two rows (objects) are in the same equivalence class if their values in a given column (attribute) is within a given interval (rather than being equal as described in Equation 2). For example, consider in Table 2 the intervals $[0, 1]$ and $[6, 7]$ for attribute m_4. We can see that it generates the partition $\{\mathtt{g}_1, \mathtt{g}_2\}, \{\mathtt{g}_3, \mathtt{g}_4\}$. We call these intervals "equivalence blocks", similarly as the "tolerance blocks" described in [10]. Equivalence blocks can be either pre-defined, allowing the user to include some background knowledge in the biclustering process, or calculated on-the-fly if a number of equivalence blocks γ is specified.

4 Experiments

4.1 Partition pattern concept lattice calculation

In order to calculate the partition pattern concept lattice for a given data-table we used the AddIntent algorithm as described in [16]. We applied AddIntent over a subset of the dataset called MovieLens 100k[1] of movie ratings containing 943 users and 50 movies (out of a total of 1682) using the predefined set of equivalence blocks $[1, 2][3, 3][4, 5]$. The dataset contains user ratings for movies

[1] `http://grouplens.org/datasets/movielens/`

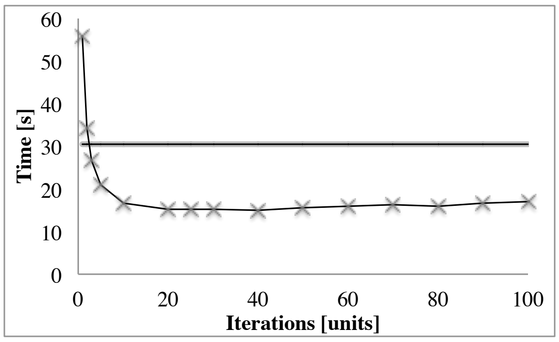

Fig. 1: AddIntent Iterations per prune vs Execution time

which range from 1 to 5. When information is not available, the matrix contains 0 which we disregard (we do not mine biclusters with columns equal to 0). The dataset contained 16532 similar column biclusters.

Empirical results showed that less than 20% of the pp-concepts within the pp-lattice actually hold a maximal bicluster. In order to improve the efficiency of AddIntent for biclustering purposes we have included a pruning step between a certain number of AddIntent iterations (each time a new intent is added to the lattice). The pruning step consists of removing from the lattice any concept that do not hold a maximal bicluster. Figure 1 shows experimental results in this regard. The graphic shows the execution time (y axis) taken by AddIntent to calculate the 16532 biclusters when a pruning step was included in a given number of iterations (x axis). The solid horizontal line represents the execution time without pruning (30.5 seconds). While initially, the execution time doubles the non-optimized version (for a lattice prune each AddIntent iteration), later the time quickly stabilizes around half the time the non-optimized version. Best time is found for 40 iterations (15 seconds).

The pruning affects the number of intent intersections performed by AddIntent. When the lattice is pruned, there are not as many intents to intersect as there were originally. However, pruning the lattice is an expensive task and adds overhead to the algorithm. The correct balance of this trade-off leads to dramatic improvements in the performance (twice in the experiments), however further experimentation in different numerical data-tables are needed to draw more conclusions regarding its setting.

4.2 Biclusters quality

A second experiment was performed over an example dataset provided with the system BicAt[2] containing 419 objects and 70 attributes. We measure the performance of our approach mining similar row biclusters compared with Cheng and Church's algorithm (CC) [3]. CC tries to find a determined number of biclusters

[2] `http://www.tik.ee.ethz.ch/sop/bicat/`

with a maximum threshold for the mean squared error δ. Results are shown in Table 4. Parameters for pp-lattice are number of equivalence blocks γ and minimal number of columns in the cluster σ. CC was executed as provided by BicAt and other parameters were left as system's default.

	Time [s]	Biclusters [Kunits]	Parameters	MSE Max	Max Size [cells]
PPL	451	901	γ=20, σ=10	0.016	209
PPL	27	36	γ=10, σ=30	0.032	372
PPL	306	390	γ=10, σ=25	0.037	442
PPL	3,404	4,471	γ=10, σ=20	0.041	462
PPL	253	314	γ=5, σ=50	0.259	1,173
CC	418	1	$\delta = 0.5$	3.2	17,752
CC	416	1	$\delta = 0.3$	2.81	17,752
CC	4,018	10	$\delta = 0.1$	4.92	17,752

Table 4: Comparison between CC and pp-lattice bicluster algorithm.

Results show a general better performance of our approach which is able to mine more than four million maximal biclusters from the dataset in less time than CC calculates only ten thousands. In terms of minimal squared error (MSE), our approach gets smaller scores which induces better quality biclusters. CC is able to find larger biclusters compared to our approach given the top-down strategy which implements. While larger biclusters can be found with our approach by decreasing the number of equivalent classes (γ), this is done at the cost of increasing the MSE as shown in Table 4. Compared to CC, our approach is better on finding many high quality and rather small biclusters inducing specialized associations among objects. CC is better at creating a global map of the entire data-table by finding larger biclusters.

5 Conclusions and research perspectives

In this work we have presented a novel technique for exhaustive similar row/column value biclustering based on FCA algorithms using partition pattern structures. We have shown the capabilities of the technique which is able to find a large number of high quality biclusters. Furthermore, biclusters are provided with an overlapping hierarchy based on a concept lattice structure. How to leverage current biclusters analysis techniques using the concept lattice is still a matter of research.

Partition pattern structures were initially proposed for functional dependencies mining [1] using association rules from pp-concepts. How these techniques may benefit from the current approach and the opposite, is an interesting subject which should be explored. Using other techniques of formal concept selection and filtering, and their associations with biclusters is another compelling aspect for a future work.

References

1. Jaume Baixeries, Mehdi Kaytoue, and Amedeo Napoli, 'Characterizing functional dependencies in formal concept analysis with pattern structures', *Annals of Mathematics and Artificial Intelligence*, (2014).
2. Jérémy Besson, Céline Robardet, Luc Raedt, and Jean-François Boulicaut, 'Mining bi-sets in numerical data', in *Knowledge Discovery in Inductive Databases*, (2007).
3. Yizong Cheng and George M. Church, 'Biclustering of expression data', in *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*, (2000).
4. Adelaide Valente Freitas, Wassim Ayadi, Mourad Elloumi, Joséluis Oliveira, Joséluis Oliveira, and Jin-Kao Hao, *Survey on Biclustering of Gene Expression Data*, 2013.
5. Bernhard Ganter and Sergei O. Kuznetsov, 'Pattern Structures and their projections', *Conceptual Structures: Broadening the Base*, (2001).
6. Bernhard Ganter and Rudolf Wille, *Formal Concept Analysis*, mathematic edn., 1999.
7. Mehdi Kaytoue, Sergei O. Kuznetsov, Juraj Macko, and Amedeo Napoli, 'Biclustering meets triadic concept analysis', *Annals of Mathematics and Artificial Intelligence*, (2013).
8. Mehdi Kaytoue, Sergei O. Kuznetsov, and Amedeo Napoli, 'Biclustering numerical data in formal concept analysis', in *Formal Concept Analysis*, (2011).
9. Mehdi Kaytoue, Sergei O. Kuznetsov, and Amedeo Napoli, 'Revisiting numerical pattern mining with formal concept analysis', *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence*, (November 2011).
10. Sergei O. Kuznetsov, 'Galois connections in data analysis: Contributions from the soviet era and modern russian research', in *Formal Concept Analysis*, volume 3626 of *Lecture Notes in Computer Science*, (2005).
11. Sergei O Kuznetsov and Sergei Obiedkov, 'Comparing Performance of Algorithms for Generating Concept Lattices', *Journal of Experimental and Theoretical Artificial Intelligence*, (2002).
12. Sara C. Madeira and Arlindo L. Oliveira, 'Biclustering algorithms for biological data analysis: A survey', *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, (January 2004).
13. Sadaaki Miyamoto, 'Lattice-valued hierarchical clustering for analyzing information systems', in *Rough Sets and Current Trends in Computing*, volume 4259 of *Lecture Notes in Computer Science*, (2006).
14. Gaurav Pandey, Gowtham Atluri, Michael Steinbach, Chad L. Myers, and Vipin Kumar, 'An association analysis approach to biclustering', in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (2009).
15. Gianvito Pio, Michelangelo Ceci, Corrado Loglisci, Domenica D'Elia, and Donato Malerba, 'A novel biclustering algorithm for the discovery of meaningful biological correlations between mirnas and mrnas', *EMBnet.journal*, **18**(A), (2012).
16. Dean van der Merwe, Sergei Obiedkov, and Derrick Kourie, 'Addintent: A new incremental algorithm for constructing concept lattices', in *Concept Lattices*, (2004).