Efficient Error Setting for Subspace Miners

Eran Shaham[1], David Sarne[1], and Boaz Ben-Moshe[2]

[1] Dept. of Computer Science, Bar-Ilan University, Ramat-Gan, 52900 Israel,
{erans,sarned}@macs.biu.ac.il
[2] Dept. of Computer Science, Ariel University, Ariel, 44837 Israel,
benmo@ariel.ac.il

Abstract. A typical mining problem is the extraction of patterns from subspaces of multidimensional data. Such patterns, known as a biclusters, comprise subsets of objects that behave similarly across subsets of attributes, and may overlap each other, i.e., objects/attributes may belong to several patterns, or to none. For many miners, a key input parameter is the error used which greatly affects the quality, quantity and coherency of the mined clusters. As the error is dataset dependent, setting it demands either domain knowledge or some trial-and-error. The paper presents a new method for automatically setting the error to the value that maximizes the number of clusters mined. This error value is strongly correlated to the value for which performance scores are maximized. The correlation is extensively evaluated using six datasets, two mining algorithms, seven prevailing performance measures, and compared with five prior literature methods, demonstrating a substantial improvement in the mining score.

Keywords: Biclustering, Subspace Mining, Error Setting

1 Introduction

In recent years we have witnessed an ever increasing availability of information associated with people, the environment, machines and their interactions. In order to benefit from such huge amounts of data, efficient data mining and analysis tools are required. Such tools can facilitate uncovering hidden regulatory mechanisms governing the data objects. Take, for example, location-aware devices (e.g., smartphone, GPS, RFID) that leave behind electronic trails. Many commercial services (e.g., Google Latitude, Microsoft GeoLife, Facebook Places and others) collect the data in order to maintain location-based social networks (LBSN). The LBSN are later used for personal marketing purposes, which are based on finding similar patterns of behavior in other users.

A typical mining problem is the extraction of patterns from subspaces of multidimensional data. Such patterns would usually be noisy and overlap with each other, i.e., objects/attributes may belong to several patterns, or to none. A common way to represent a dataset is by a matrix of values, where the rows represent objects, the columns represent attributes, and the data entries are

the measurements of the objects over the attributes [12]. Early mining techniques used the key concept of mining patterns (clusters) formed by a subset of the objects over *all* attributes, or vice versa [12, 4]. Seminal work by Cheng and Church [8] in the area of gene expression data, led to focusing on mining biclusters: a *subset* of objects that exhibit similar behavior across a *subset* of attributes, or vice versa. This problem, which for the most interesting cases was proved to be NP-complete [22], has attracted the attention of many researchers. This is mostly due to its extensive applicability, yielding various bicluster mining algorithms (miners) [12, 16, 20].

One key parameter that affects the ability to mine effective biclusters is the error allowed. This parameter is commonly used in models of biclusters. Its primary goal is to deal with the inherent noise that characterizes real-life datasets (e.g., human or machine error), allowing some degree of error within the mined patterns. Taking an error of zero will lead to biclusters with exact matching (mostly used in string matching), while taking an error of 100% will encapsulate the entire dataset (i.e., contain all objects and attributes) which is futile in most cases. As such, this is the main parameter through which the user of a mining algorithm can control the quality of the mined clusters. Thus, correctly setting the error is of high importance. Since error is a feature of the dataset itself, the preferred value to be used when mining biclusters is dataset dependent [1, 7], and testing the fit of a suggested value requires domain knowledge. The problem of setting the error is particularly challenging as in most cases the process is unsupervised. This means that the user is interested in finding the patterns in the data, but cannot a priori estimate whether a mined cluster is indeed the result of a regulatory mechanism in the data or a random effect.

Despite the great importance of this parameter, very little attention has been given to the way the error should be set for bicluster mining. Most prior work assumes the miner's user is capable of evaluating the mined clusters (e.g., has extensive domain knowledge or a way of validating the mined patterns) and thus is capable of suggesting trial-and-error methods for setting the error [15, 33]. Other prior work suggests rules-of-thumb, such as setting the error value to 5% of the dataset range [19] or using the smallest value possible which still yields patterns [7]. Very few authors have suggested heuristic methods for setting the error [25, 32]. Heuristic methods, as explained in detail in the next section, rely on an exogenous parameter thus posing a different problem of how to set the value of these new parameters.

In this paper we suggest an unsupervised method for automatically setting the error for bicluster miners. The method suggests using the *error value* which maximizes the number of the mined clusters. We evaluated our method extensively using six datasets of different domains, two state-of-the-art bicluster mining algorithms, and seven prevailing performance measures that evaluate the relevance of the mined biclusters. The results show that our method scores close to the maximum value for each of the above configurations. In addition, we evaluated our method in comparison to five error-setting methods that have been proposed in prior literature. The results suggest that in the majority of cases,

our method performs substantially better in terms of the performance measures, while in a small number of cases it results in a slight degradation. Overall, our method successfully releases the user from the task of setting the error parameter value, enabling effective mining even when the dataset properties are not fully known.

The remainder of the paper is structured as follows. The next section formally introduces and reviews the biclustering model, different approaches to mining biclusters (miners), the role of the error parameter value in the process, and prior approaches for setting the value of that parameter. Section 3 presents the main hypothesis and the reasoning regarding the correlation between the performance measure score and the number of clusters mined, as a function of the error value. Section 4 presents experimental design, including a detailed description of the datasets that were used, the main measures and their interpretation, and the mining algorithms with which biclusters were mined. Section 5 presents the results for validating the main hypothesis and their analysis. Subsection 5.1 presents the performance achieved for the various performance measures. The results of the comparison with prior methods for setting the error value are given in Subsection 5.2. Finally, conclusions and directions for future research are outlined in Section 6. Related work is cited whenever applicable throughout the paper.

2 Models, Error-setting Methods, and Performance Measures

Bicluster mining belongs to a class of problems where the goal is the extraction of patterns from subspaces of multidimensional data, where the rows represent objects, the columns represent attributes and the data entries are the measurements of the objects over the attributes [12]. Early mining techniques sought for a fully dimensional cluster: a subset of the objects over *all* attributes (or vice versa) exhibiting such a pattern [12]. However, when mining high-dimensional datasets, these methods become ineffective and inaccurate due to the "curse of dimensionality" [5].

Cheng and Church [8], in their seminal paper in the field of gene expression data, introduced a novel technique for mining clusters which focuses on mining **biclusters**: a *subset* of the objects over a *subset* of the attributes. Their approach was soon followed by many researchers (see surveys by [3, 12, 16]), applying a variety of mining techniques: greedy, projected clustering, exhaustive enumeration, spectral analysis, CTWC, bayesian networks, etc. The model they used, which is still the most common, is defined as follows.

Definition 1. *Let A be an $m \times n$ real number matrix of O objects ($|O| = m$) and T attributes ($|T| = n$). A bicluster of A, denoted (I,J), is a sub-matrix A_{IJ}, where I is a subset of the rows ($I \subseteq O$) and J is a subset of the columns ($J \subseteq T$).*

The desired bicluster is one which the sub-matrix elements exhibit some predefined pattern. As real world datasets are generally noisy, mining algorithms

usually allow for some deviation from the "perfect pattern". For this purpose, the biclustering models are usually extended to include some similarity measure, which measures the deviation of the mined biclusters from the "perfect pattern", i.e., the "error" [6]. Commonly, the degree of deviation is constrained by a maximum allowed value, namely, the similarity measure score of the mined patterns would be lower than some pre-defined error $\delta \geq 0$. The choice of one similarity measure over the other would usually affect the nature of the mined clusters [30]. Next, we review several prevalent similarity measures found in literature.

2.1 Similarity Measures

Due to the "curse of dimensionality" [5], well-known distance functions (e.g., Euclidean distance, Manhattan distance, cosine distance and others) are commonly not adequate as similarity measures when mining multi-dimensional data [5, 31]. To overcome this shortcoming, new similarity measures have been developed over the years. Among these, the two most commonly used are the *mean squared residue* and *pCluster*.

Mean Squared Residue The *mean squared residue* measure was first used in the field of gene expression analysis by Cheng and Church [8] to assess the correlation between genes and conditions within a bicluster, and was soon adopted by many researchers [1, 6]. The measure is defined as follows.

$$H(I,J) = \frac{1}{|I||J|} \sum_{i \in I, j \in J} (a_{ij} - a_{iJ} - a_{Ij} + a_{IJ})^2 \tag{1}$$

where a_{ij} is the element corresponding to the i^{th} row and the j^{th} columns of A and:

$$a_{iJ} = \frac{1}{|J|} \sum_{j \in J} a_{ij},\ a_{Ij} = \frac{1}{|I|} \sum_{i \in I} a_{ij},\ a_{IJ} = \frac{1}{|I||J|} \sum_{i \in I, j \in J} a_{ij},$$

are the row and column means and the mean of the submatrix (I,J), respectively. A sub-matrix A_{IJ} is called a δ-bicluster if the value of $H(I,J)$ is lower than some error value, i.e., $H(I,J) \leq \delta$ for an error $\delta \geq 0$.

The mean squared residue measure suffers from the following shortcomings: (i) the score of a bicluster can be smaller than the score of a sub-matrix of this bicluster. Therefore, a sub-cluster of a bicluster is not necessarily a bicluster [19, 31, 33]; (ii) the score is insensitive to outliers, i.e., biclusters may differ only by the few outliers they contain [19, 31]; (iii) the score is biased toward flat biclusters with low row variance. This may result, in the domain of gene expression, in mining of biclusters which are less interesting as they contain genes with relatively unfluctuating expression levels [6]; and (iv) the relation of the score to a biological regulatory model is unclear [1, 19]. These weaknesses have led in recent years to the emergence of an alternative measure known as *pCluster*.

pCluster The *pCluster* similarity measure [23, 31, 33, 35] is a score defined over a sub-matrix of 2×2. The concept is to find small sub-matrices, each of which is a δ-bicluster, to serve as building-blocks. Such building-blocks would then be assembled into a larger δ-bicluster. The definition of the measure is given as follows. Let $p, q \in I$ and $j, k \in J$, then:

$$pScore\left(\begin{bmatrix} A_{pj} & A_{pk} \\ A_{qj} & A_{qk} \end{bmatrix}\right) = |(A_{pj} - A_{qj}) - (A_{pk} - A_{qk})|. \tag{2}$$

The sub-matrix A_{IJ} forms a δ-bicluster if for any 2×2 sub-matrix X in A_{IJ}, we have $pScore(X) \leq \delta$ for some error $\delta \geq 0$. Alternatively, for every $p, q \in I$ and for every $j, k \in J$:

$$|(A_{pj} - A_{qj}) - (A_{pk} - A_{qk})| \leq \delta. \tag{3}$$

Intuitively, the meaning of the pScore measure is that the changes of values in the sub-matrix are limited by δ. It is worth noting that the pScore metric can be measured either horizontally or vertically as the inequality is equivalent:

$$\begin{aligned} & |(A_{pj} - A_{qj}) - (A_{pk} - A_{qk})| && (4)\\ = \ & |(A_{pj} - A_{pk}) - (A_{qj} - A_{qk})| && (5)\\ = \ & \left(\begin{bmatrix} A_{pj} & A_{qj} \\ A_{pk} & A_{qk} \end{bmatrix}\right). && (6) \end{aligned}$$

Over the years, several enhancements have been offered to the δ-bicluster definition. These extend the *pCluster* error model by detecting *larger* δ-bicluster building blocks. Guan et al. [10] use the following δ-bicluster definition. Let k be an arbitrary attribute ($k \in J$). The sub-matrix A_{IJ} is a δ-bicluster if for every $p, q \in I$:

$$\max_{j \in J}(A_{pj} - A_{pk}) - \min_{j \in J}(A_{qj} - A_{qk}) \leq \delta/2. \tag{7}$$

In a similar manner, a sub-matrix A_{IJ} is defined by Wang et al. [30] (see example in Fig. 1, based on [30]) to be a δ-bicluster if for every $p, q \in I$:

$$\forall j \in J, \ |(A_{pj} - A_{qj}) - (A_{pk} - A_{qk})| \leq \delta/2. \tag{8}$$

Fig. 1 depicts the reasoning behind the above inequality. Let $J = \{k, b, c, d, e, f\}$. The sub-matrix A_{IJ} would be a δ-bicluster if for every $p, q \in I$, the distance between p and q, with respect to dimension k, is $\leq \delta/2$. The distance between p and q on each column $j \in J$ would be $\leq \Delta \pm \delta/2$, where Δ is the distance between p and q on column $k \in J$.

Procopiuc et al. [25] and Califano et al. [7] use the following δ-bicluster definition. A sub-matrix A_{IJ} is a δ-bicluster if for every $p, q \in I$:

$$\forall j \in J, \ \max_{p \in I}(A_{pj}) - \min_{q \in I}(A_{qj}) \leq \delta/2. \tag{9}$$

Liu et al. [15] and Melkman et al. [19] define a sub-matrix A_{IJ} to be a δ-bicluster if for every $p, q \in I$:

$$\forall j \in J, \ |A_{pj} - A_{qj}| \leq \delta/2. \tag{10}$$

The above definitions of the *pCluster* error model are equivalent. A formal proof is available at supporting webpage [28].

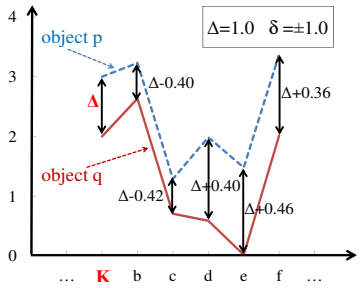

Fig. 1: An illustration of Wang et al. similarity measure [30]. For Δ=1 and δ=1: $\forall j \in J,\ |(A_{pj} - A_{qj}) - (A_{pk} - A_{qk})| \leq \delta/2$.

2.2 Existing Methods for Error Setting

One key parameter bicluster mining algorithms use, which has a critical effect over the results obtained is the *error*. The mining algorithm would supposedly report all biclusters which comply with the input error. Setting the error too low would usually yield flat (e.g., constant value) biclusters which are of an insignificant contribution. Setting the error too high might result in a single futile bicluster covering the entire dataset. The best error is usually dataset dependent [1, 7], hence setting it correctly without prior domain knowledge is not trivial.

Many papers suggest a trial-and-error method for setting the error. This is usually done by testing a variety of error values and choosing the one which presents superior results [15, 33]. As the mining of biclusters is usually an unsupervised process, such a way of evaluating the results is commonly unavailable. As a result, it is common to encounter different authors using different errors for the same dataset. For example, using a similar model of error, for the yeast dataset [8] containing 8,224 yeast genes (objects) and 17 conditions (attributes), Liu et al. [15] found that the best error to use is δ=0.1, Yoon et al. [33] found an error of δ=1.0 to be preferable, while Guan et al. [10] found the error of δ=10.0 to be superior.

Melkman et al. [19] suggest a rule-of-thumb for setting the error to a value of 5% of the dataset range. Califano et al. [7] recommend setting the error to the smallest value possible which still yields patterns.

Procopiuc et al. [25] suggest the following technique. Let p^i be a data point and q^i its nearest neighbor ($p^i, q^i \in O$, $1 \leq i \leq m = |O|$). Let $\delta^i = \frac{1}{n}\sum_{j=1}^{n} |p^i_j - q^i_j|$ be the average distance between p^i and q^i. Then, choose the value of the error to be $\delta = C \cdot \frac{1}{m}\sum_{i=1}^{m} \delta^i$ for some small constant C. The drawbacks of this technique are: (i) the calculation of the average distance between the points p^i and q^i uses all attributes. The inclusion of noisy irrelevant attributes would result in an inaccurate measure; (ii) using all attributes is futile as the distance between any two multi-dimensional points tends to be the same [5]; (iii) the heuristic specifies the use of an unknown constant C, of which the user has no knowledge regarding its setting; and (iv) the constant C is arbitrary, i.e., forcing the user to specify

the unknown value of C only transforms the problem of setting the unknown error δ into that of setting the unknown value C.

Yiu et al. [32] present the following technique. For each dimension j ($j \in T$), sort the values of A_{ij} ($i \in O$). Then, slide a window of size $\alpha|O|$, $0.0 \leq \alpha \leq 1.0$, along the values of the sorted set. For each position of the window, compute the value difference between the first and the last element of the window. Let δ^j be the average of the differences over all window positions. Then, set the error as the average over all δ^j, i.e., $\delta = \frac{1}{n}\sum_{j=1}^{n}\delta^j$. The disadvantage of the technique lies in its use of the unknown parameter α, which sets the size of the sliding window. In order for the user to set the unknown α, some rule-of-thumb or trial-and-error is needed.

2.3 Performance Measures

In order to evaluate the success of the bicluster algorithms, with the error set according to the different approaches, we use external performance measures. These measures characterize the quality, accuracy and relevance of the mined patterns (see survey in [9, 11, 21, 26]). Among the main properties emphasized by these measures are: (i) object criterion – a mined cluster should only obtain the cluster's objects; (ii) attribute criterion – a mined cluster should only obtain the cluster's attributes; (iii) redundancy criterion – a cluster should only be identified and reported once (i.e., not as part of several sub-clusters); and (iv) identification criterion – all clusters should be identified.

To evaluate our method, we use the following seven popular performance measures: (i) F-score [29]; (ii) Entropy [36]; (iii) Purity [36]; (iv) Variation of Information – VI [18]; (v) Q_0 [9]; (vi) Misclassification Index – MI [34]; and (vii) Normalized Mutual Information – NMI [17]. For the computation of the above measures, we used the publicly available *ClusterEvaluator* package [26].

3 Main Hypothesis

Our main hypothesis is that *there is a strong correlation between the error values for which the number of clusters mined reaches its maximum and the error values for which the performance measure score reaches its maximum. In particular, the two reach their maximum for a similar error.*

The existence of the correlation enables the automatic setting of the error in a user-free, simple fashion manner, which does not require any specific domain knowledge nor any actual reasoning about the content of the mined clusters. In order to set the error, one should find the error in which the number of the mined clusters reaches its peak. To find that peak, one can start from a seed of zero error and adopt any of the various algorithms for finding a maximum [24], such as simplex method (also called 'amoeba' or the Nelder-Mead method) [24].[3]

[3] The simplex method is known for its simplicity, as it makes almost no assumptions regarding the function, and does not require any derivative evaluation. Furthermore,

The reasoning behind the correlation is given within the scope of precision, recall and F-score.[4] When supplied with a maximum allowed error, mining algorithms would usually produce *many* biclusters, all associated with an error lower than the maximum allowed. The increase in the allowed error has two conflicting effects related to the number of clusters mined.

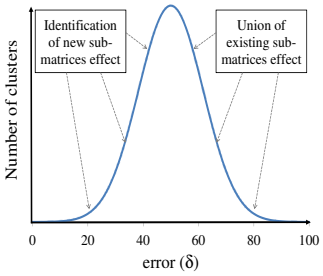

Fig. 2: Conflicting effects governing the number of mined clusters.

The first is the identification of new sub-matrices associated with an error that was prohibited by lower maximum error (i.e., an increase in the number of clusters mined; see left side of Fig. 2). The second is the union of sub-matrices that were considered separate clusters for lower maximum error, but can now be considered as one cluster associated with an error below the maximum allowed error (i.e., a decrease in the number of clusters mined; see right side of Fig. 2). When the maximum allowed error is set to a low value, miners will mine a small number of trivial flat (e.g., constant value) biclusters. In this case, when the maximum allowed error is increased, the first effect dominates the second, as the maximum error is small enough to preclude union of clusters; however, any increase in the allowed error uncovers many new subsets of the "real" clusters as legitimate clusters. As the maximum allowed error keeps increasing, the second effect becomes more dominant, as the number of clusters that can be unified due to the increase in the allowed error grows exponentially. When the error reaches its maximum value, only one cluster is mined, i.e., the cluster which contains the entire dataset.

The effect of this behavior on the F-score is as follows. For low error values, where the number of generated clusters is relatively small, the F-score is low, mainly due to the recall measure – the mined clusters are indeed precise as the tight error precludes the inclusion of irrelevant elements. However, since these

in our case of a single variable function, not only does this method converge, but it does so while using a relatively small number of function evaluations [14].

[4] The typical use of this score is through its harmonic mean, which is the weighted harmonic mean of the precision and recall, and is defined as follows: $F_1 = 2 \cdot (precision \cdot recall)/(precision + recall)$. In terms of Type-I and type-II errors it is defined as: $F_1 = (2 \cdot true\ positives)/(2 \cdot true\ positives + false\ negatives + false\ positives)$.

clusters are mostly subsets of the "real" clusters, the recall measure suffers. As the number of mined clusters increases, the recall measure improves. Nevertheless, once the effect of joining different sub-matrices due to the increase in the allowed error becomes more apparent (i.e., when the number of clusters is on the decrease) the precision measure drops sharply, resulting in a total decrease in the F-score despite the marginal improvement in the recall. Therefore, when the number of mined clusters reaches its peak, it is likely that the mining using that error offers an effective tradeoff between precision and recall, in the sense that each of the mined clusters contains most elements of the "real" cluster it represents and does not include many elements which are either noise or belong to other "real" clusters.

4 Experimental Methodology

In the following paragraphs we describe the methodology of the experiments used to demonstrate the existence of the correlation hypothesized in the previous section.

4.1 Datasets

For our experiments, we used six published datasets, comprising a varied set of examples which differ in dimension, domain, number of classes, balanced vs. unbalanced classes (membership-wise), and natures (i.e., temporal vs. spatial). Each of the datasets contains labeled objects, which are used to evaluate (post factum) the mined clusters. The datasets are freely available (see the supporting webpage [28]).

The *Sonar* dataset [2] comprises 208 objects of sonar signals bounced off two object classes: metal cylinders (53%) and cylindrical rocks (47%). The dataset comprises 60 time readings as attributes. The *Mixed Bag* dataset [13] comprises the contour lines of 160 objects of nine shape classes: bone (12.5%), glass (12.5%), cup (12.5%), device (12.5%), fork (12.5%), pencil (12.5%), hand (7%), rabbit (7.5%), and tool (10.5%). The dataset comprises 1614 contour sampling as attributes. The *Waveform* dataset [2] comprises the synthetic generated waves of 5000 objects of three functions: function #1 (33.2%), function #2 (32.9%), and function #3 (33.9%). The dataset comprises 40 time readings as attributes. The *Arrhythmia* dataset [2] comprises the cardiac arrhythmias of 452 objects of 13 classes: normal (54%), ischemic (10%), anterior myocardial (3%), inferior myocardial (3%), tachycardia (3%), bradycardia (5.5%), PVC (1%), supraventricular (0.5%), left bundle (2%), right bundle (11%), ventricular (1%), fibrillation (1%), and others (5%). The dataset comprises 279 attributes, some categorial (e.g., Sex) and others temporal (e.g., time readings). The *Shapes* dataset [13] comprises the contour lines of shapes randomly selected from five datasets of objects: arrowhead, butterfly, fish, seashell, and shields (117 randomly selected objects from each of the five datasets). The dataset comprises 1024 contour sampling as attributes. The *Skylines* dataset [27] comprises skylines as viewed

by various random observers on a topographic map. The dataset comprises 10 classes, each with 20 objects (observers), and 720 z-blocking angles as attributes.

4.2 Mining Algorithms

For the reasons considered in Section 2.1 we preferred the *pCluster* measure as the similarity measure to be used by the mining algorithms employed in our experiments. Consequently, our experiments used the following two widespread biclustering algorithms: *pCluster* [31], and *DOC* [25]. We chose these two biclustering algorithms as representatives of two different mining approaches.

The *pCluster* algorithm is deterministic in the sense that it discovers all qualified biclusters. The algorithm uses a depth-first search to simultaneously mine multiple biclusters. As the problem is NP-complete, the run-time of the algorithm may be exponential (although pruning techniques used in the early stages of the algorithm greatly reduce the overall complexity in practice). The *pCluster* algorithm executable was obtained from the author's website [31].

The *DOC* algorithm is probabilistic in the sense that it discovers approximations of the qualified biclusters. The algorithm uses a projected clustering approach. The use of the Monte-Carlo technique allows, with high probability, the mining of 2-approximated optimal biclusters. The algorithm has a polynomial time complexity. We have implemented the *DOC* algorithm ourselves based on [25].[5]

Both algorithms require, in addition to the maximum allowed error, a density criteria in the form of a minimum number of objects and a minimum number of attributes. A valid bicluster thus will need to have at least the specified number of rows and columns.

4.3 Experimental Design

The experiments were designed in the following way. Each of the mining algorithms (see Subsection 4.2) was run on each of the datasets (see Subsection 4.1) while setting the error to a variety of values. In order not to miss small biclusters (membership-wise), the minimum number of objects was set to two. To reduce the chance of mining artifacts, the minimum number of attributes was set to 5% of the total [19]. For the *DOC* algorithm [25], the number of iterations used for each error value was set to 10000. Due to the probabilistic nature of the *DOC* algorithm, and to reduce the effect of randomness on the results, we repeated the experiment with *DOC* and averaged the results of each error configuration for 10 trials.

For each of the above configurations, we recorded the mined biclusters, and computed the corresponding score according to each of the performance measures (see Subsection 2.3). Based on the common practice with mining algorithms [4],

[5] The implementation was of the *DOC* algorithm and not of the *FastDOC* algorithm as the probabilistic guarantee of mining the optimal biclusters was more important than the reduction in the run-time complexity.

each bicluster was assigned to the class which is most frequent in the cluster (i.e., plurality, relative majority). In the cases where a cluster can be attributed to several classes (i.e., no plurality arises), it was randomly ascribed to one of the classes. For each class and for each performance measure, we choose the bicluster which obtained the best score. For each performance measure, the score of a run (a tested error and a given dataset) was taken to be the average score over all classes.

5 Results

Next, we present the results of the experiments. Subsection 5.1 summarize the performance achieved by the proposed error-setting heuristic, presenting persistent high scores across various configurations. Subsection 5.2 presents a comparison of the results obtained with our method and those obtained by setting the error according to methods suggested in prior literature, demonstrating a substantial improvement in the mining score.

5.1 Heuristic Performance

Table 1 and Table 2 present the performance achieved by the proposed error-setting heuristic when using the *DOC* and *pCluster* miners, respectively. Each result represents the ratio between the value obtained according to the specific measure when using the number-of-clusters-maximizing error and the maximum value that can be achieved according to that measure.

Table 1: Scores Obtained for *DOC* Miner.

	%max{score}						
Dataset	MI	Entropy	VI	Purity	Q_0	F_1	NMI
Sonar	86%	100%	95%	85%	94%	99%	99%
Mixed Bag	100%	99%	100%	94%	95%	74%	72%
Waveform	94%	100%	100%	87%	97%	96%	100%
Arrhythmia	100%	77%	45%	69%	43%	94%	94%
Shapes	99%	100%	95%	97%	99%	95%	100%
Skylines	100%	93%	99%	100%	87%	86%	74%
average	96%	95%	89%	89%	86%	91%	90%

Tables 1 and 2 show that, other than four exceptions (the VI and Q_0 scores for the Arrhythmia dataset with the *DOC* miner and the *Purity* and F_1 scores for the Skylines and Shapes dataset, respectively, with the *pCluster* miner), setting the error to be equal to the number-of-clusters-maximizing error results in performance close to that achieved with the optimal error, for all the additional measures and with both miners. Furthermore, the tables show that even in those

Table 2: Scores Obtained for *pCluster* Miner.

	%max{score}						
Dataset	MI	Entropy	VI	Purity	Q_0	F_1	NMI
Sonar	100%	100%	100%	100%	96%	99%	100%
Mixed Bag	93%	99%	91%	98%	92%	99%	98%
Waveform	92%	100%	100%	95%	100%	100%	100%
Arrhythmia	100%	76%	100%	86%	89%	91%	100%
Shapes	93%	100%	85%	83%	96%	40%	84%
Skylines	90%	100%	82%	46%	98%	89%	71%
average	95%	96%	93%	85%	95%	86%	92%

very few cases where a poor F-score was obtained when using the number-of-clusters-maximizing error (i.e., in the case of the Mixed Bag dataset with the *DOC* miner and Shapes dataset with the *pCluster* miner) the performance results according to the other measures are very encouraging, indicating that the appropriate clusters were mined and that the low score according to the specific measure is probably a result of a unique inherent feature of the dataset.

Obviously, in order to get into the roots of how the number of mined clusters are correlated with any of these measures one needs to investigate how their different components are influenced, similarly to the way the effect over precision and recall was analyzed for the F-score in Section 3. While this is beyond the scope of the current paper, we stress that overall all these measures emphasize properties that are likely to be found within the optimal set of clusters and as such have strong correlation with the precision and recall. The fact that the same qualitatively positive results were obtained with all these additional measures substantially strengthen our confidence in the appropriateness of using the number-of-clusters-maximizing error as a default error value for biclustering miners.

5.2 Comparison with Alternative Methods

In order to compare the performance of our method with the ones used in literature we conducted the following experiment. We ran the two miners (*DOC* and *pCluster*; see Subsection 4.2) across six datasets (see Subsection 4.1). For each pair, we calculated the ratio between the F-score achieved when using our method and the one achieved when using any of the five alternative methods (see Subsection 2.2). For example, a ratio of 2 means that our method achieved twice the percentage of the maximum F-score than the alternative technique, while a ratio of 0.9 means that our method achieved 90% of the score achieved by the alternative technique.

To ease readability, we present only the results of the performance comparison between our method and the one of Yiu et al. [32], using the *DOC* miner, each of the six datasets, and various settings for the *unknown* $0.0 \leq \alpha \leq 1.0$ parameter (Table 3). The results for all other configurations (including the ones of the

pCluster miner) are even better, and are given in details in the supporting webpage [28].

Table 3: Suggested Method vs. Yiu et al. Method.

Dataset \ α	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Sonar	2.00	1.14	1.01	1.00	1.01	1.02	1.04	1.04	1.84	1.08
Shapes	3.38	1.65	1.17	1.00	0.96	0.95	0.98	1.05	1.25	2.11
Arrhythmia	1.67	1.67	1.67	1.67	1.67	1.67	1.67	1.67	1.67	1.67
Waveform	9.64	5.27	1.76	1.15	1.00	0.96	0.96	0.98	1.02	1.42
Mixed Bag	1.12	0.80	0.76	0.74	0.76	0.79	0.88	1.05	1.65	2.30
Skylines	1.01	0.86	0.86	0.89	1.00	1.00	1.14	1.19	1.31	2.41

The important finding of Table 3 is that for most value settings, our method surpassed the alternative technique. Furthermore, in the cases in which it did not, the achieved score was still high (88% on average) in comparison to the maximum score achieved by the alternative technique. The fact that a similar pattern was obtained for all other configurations further strengthens the results, and corroborates that the use of our method is preferable: not only is the user free from setting an *unknown* parameter, but also, in most cases, the achieved score is higher.

6 Discussion and Conclusions

The importance of mining clusters is unquestionable and has been thoroughly discussed and demonstrated in cited prior work. Similarly, the extensive biclustering literature includes many examples of the benefit of mining biclusters as opposed to traditional approaches. In order to identify a regulatory mechanism within a noisy dataset, a bicluster miner must rely on some degree of error. Setting the error parameter properly is crucial for the mining of coherent biclusters.

The paper presents a hypothesis regarding the correlation between the number of clusters and the errors for which performance scores are maximized. This hypothesis, which has important implications for setting the error parameter, is tested using extensive experimentation that spreads over six datasets, two state-of-the-art bicluster mining algorithms, and evaluated by seven prevailing performance measures. The encouraging results reported in Section 5 experimentally validate the correlation hypothesized. This enables setting the error parameter value automatically in an unsupervised, relatively simple manner. One needs only to follow the number of clusters mined for each error value and pick the one for which the number of mined clusters reaches its peak. As emphasized throughout the paper, this does not require any specific domain knowledge nor any actual reasoning about the content of the mined clusters. Furthermore, to find that peak, one can use various algorithms, as discussed in Section 3.

The findings reported are of particular importance in light of the relatively poor performance of existing methods for setting the error.

The new method does not always result in the best error value; however, the performance achieved is very close to that of the performance-maximizing error. This is demonstrated with a set of measures that have been used in prior literature for evaluating the relevance of mined biclusters. Furthermore, the comparison with five existing error-setting methods reveals that, in the majority of cases, the new method results in substantially better performance. For a small number of cases it performs slightly worse than the competitor. These results were obtained for both algorithms used.

Finally, we note that the hypothesis tested in this paper may not hold when the mining is performed by an algorithm that uses an error model in which a sub-matrix of a bicluster is not necessarily a bicluster in itself (e.g., *mean squared residue*) and in cases of "incorrect" class labeling of the objects. The adaptation of the method for such cases is thus left for future research. Other future research that can benefit from the findings reported in this paper includes extensions to more complex biclustering models such as those that consider shifts (addition by a constant) and scales (multiplication by a constant), lag and fuzziness.

Acknowledgments. We are thankful to Haixun Wang for providing us with the source code of the *pCluster* algorithm [31]. Also, we are grateful to Andrew Rosenberg for the permission to use the *ClusterEvaluator* package [26], and for enlightening remarks.

References

1. Aguilar-Ruiz, J.S.: Shifting and scaling patterns from gene expression data. Bioinformatics 21(20), 3840–3845 (2005)
2. Bache, K., Lichman, M.: UCI Machine Learning Repository (2013)
3. Berkhin, P.: A survey of clustering data mining techniques. Grouping Multidimensional Data pp. 25–71 (2006)
4. Berson, A., Smith, S., Thearling, K.: Building data mining applications for CRM. McGraw-Hill New York (2000)
5. Beyer, K., Goldstein, J., Ramakrishnan, R., Shaft, U.: When is "nearest neighbor" meaningful? ICDT pp. 217–235 (1999)
6. Bryan, K., Cunningham, P.: Bottom-up biclustering of expression data. In: CIBCB. pp. 1–8 (2006)
7. Califano, A., Stolovitzky, G., Tu, Y.: Analysis of gene expression microarrays for phenotype classification. In: ISMB. vol. 8, pp. 75–85 (2000)
8. Cheng, Y., Church, G.M.: Biclustering of expression data. In: ISMB. pp. 93–103 (2000)
9. Dom, B.E.: An information-theoretic external cluster-validity measure. In: UAI. pp. 137–145 (2002)
10. Guan, J., Gan, Y., Wang, H.: Discovering pattern-based subspace clusters by pattern tree. KBS 22(8), 569–579 (2009)
11. Günnemann, S., Färber, I., Müller, E., Assent, I., Seidl, T.: External evaluation measures for subspace clustering. In: CIKM. pp. 1363–1372 (2011)

12. Jiang, D., Tang, C., Zhang, A.: Cluster analysis for gene expression data: A survey. TKDE 16(11), 1370–1386 (2004)
13. Keogh, E., Wei, L., Xi, X., Lee, S.H., Vlachos, M.: LB_Keogh supports exact indexing of shapes under rotation invariance with arbitrary representations and distance measures. In: VLDB. pp. 882–893 (2006)
14. Lagarias, J., Reeds, J., Wright, M., Wright, P.: Convergence Properties of the Nelder–Mead Simplex Method in Low Dimensions. SIOPT 9(1), 112–147 (1998)
15. Liu, G., Sim, K., Li, J., Wong, L.: Efficient mining of distance-based subspace clusters. SADM 2(5-6), 427–444 (2009)
16. Madeira, S.C., Oliveira, A.L.: Biclustering algorithms for biological data analysis: a survey. TCBB 1(1), 24–45 (2004)
17. McDaid, A.F., Greene, D., Hurley, N.: Normalized mutual information to evaluate overlapping community finding algorithms. CoRR abs/1110.2515 (2011)
18. Meilă, M.: Comparing clusterings—an information based distance. J. Multivar. Anal. 98(5), 873–895 (2007)
19. Melkman, A.A., Shaham, E.: Sleeved CoClustering. In: KDD. pp. 635–640 (2004)
20. Moise, G., Zimek, A., Kroeger, P., Kriegel, H., Sander, J.: Subspace and projected clustering: experimental evaluation and analysis. KAIS 21(3), 299–326 (2009)
21. Patrikainen, A., Meila, M.: Comparing subspace clusterings. TKDE 18(7), 902–916 (2006)
22. Peeters, R.: The maximum edge biclique problem is NP-complete. DAM 131(3), 651–654 (2003)
23. Pei, J., Zhang, X., Cho, M., Wang, H., Yu, P.S.: Maple: A fast algorithm for maximal pattern-based clustering. In: ICDM. pp. 259–266 (2003)
24. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical recipes in C: the art of scientific computing. Cambridge University Press (1992)
25. Procopiuc, C.M., Jones, M., Agarwal, P.K., Murali, T.: A Monte Carlo algorithm for fast projective clustering. In: SIGMOD. pp. 418–427 (2002)
26. Rosenberg, A., Hirschberg, J.: V-Measure: A Conditional Entropy-Based External Cluster Evaluation Measure. In: EMNLP-CoNLL. vol. 7, pp. 410–420 (2007)
27. Shaham, E., Sarne, D., Ben-Moshe, B.: Sleeved co-clustering of lagged data. KAIS 31(2), 251–279 (2012)
28. Supporting webpage (2013), `http://tinyurl.com/Supporting-MLDM14`
29. Van Rijsbergen, C.: Information retrieval. Butterworths, 2nd edn. (1979)
30. Wang, H., Chu, F., Fan, W., Yu, P.S., Pei, J.: A fast algorithm for subspace clustering by pattern similarity. In: SSDBM. pp. 51–60 (2004)
31. Wang, H., Wang, W., Yang, J., Yu, P.S.: Clustering by pattern similarity in large data sets. In: SIGMOD. pp. 394–405 (2002)
32. Yiu, M.L., Mamoulis, N.: Iterative projected clustering by subspace mining. TKDE 17(2), 176–189 (2005)
33. Yoon, S., Nardini, C., Benini, L., De Micheli, G.: Enhanced pClustering and its applications to gene expression data. In: BIBE. pp. 275–282 (2004)
34. Zeng, Y., Tang, J., Garcia-Frias, J., Gao, G.R.: An adaptive meta-clustering approach: combining the information from different clustering results. In: CSB. pp. 276–287 (2002)
35. Zhao, L., Zaki, M.J.: TRICLUSTER: an effective algorithm for mining coherent clusters in 3D microarray data. In: SIGMOD. pp. 694–705 (2005)
36. Zhao, Y., Karypis, G.: Criterion functions for document clustering: Experiments and analysis. Machine Learning (2001)