SSC : Statistical Subspace Clustering

Laurent Candillier[1,2], Isabelle Tellier[1], Fabien Torre[1], Olivier Bousquet[2]

[1] GRAppA - Charles de Gaulle University - Lille 3
candillier@grappa.univ-lille3.fr
[2] Pertinence - 32 rue des Jeûneurs -75002 Paris
olivier.bousquet@pertinence.com

Abstract. *Subspace clustering* is an extension of traditional *clustering* that seeks to find *clusters* in different subspaces within a dataset. This is a particularly important challenge with high dimensional data where the *curse of dimensionality* occurs. It has also the benefit of providing smaller descriptions of the *clusters* found.
Existing methods only consider numerical databases and do not propose any method for *clusters visualization*. Besides, they require some input parameters difficult to set for the user. The aim of this paper is to propose a new *subspace clustering* algorithm, able to tackle databases that may contain continuous as well as discrete attributes, requiring as few user parameters as possible, and producing an interpretable output.
We present a method based on the use of the well-known *EM algorithm* on a probabilistic model designed under some specific hypotheses, allowing us to present the result as a set of rules, each one defined with as few relevant dimensions as possible. Experiments, conducted on artificial as well as real databases, show that our algorithm gives robust results, in terms of classification and interpretability of the output.

1 Introduction

Clustering is a powerful exploration tool capable of uncovering previously unknown patterns in data [3]. *Subspace clustering* is an extension of traditional *clustering*, based on the observation that different *clusters* (groups of data points) may exist in different subspaces within a dataset. This point is particularly important with high dimensional data where the *curse of dimensionality* can degrad the quality of the results. *Subspace clustering* is also more general than *feature selection* in that each subspace is local to each *cluster*, instead of global to everyone. It also helps to get smaller descriptions of the *clusters* found since *clusters* are defined on fewer dimensions than the original number of dimensions.

Existing methods only consider numerical databases and do not propose any method for *clusters visualization*. Besides, they require some input parameters difficult to set for the user. The aim of this paper is to propose a new *subspace clustering* algorithm, able to tackle databases that may contain continuous as well as discrete attributes, requiring as few user parameters as possible, and producing an interpretable output. We present a method based on the use of a probabilistic model and the well-known *EM algorithm* [14]. We add in our model

the assumption that the *clusters* follow independent distributions on each dimension. This allows us to present the result as a set of rules since dimensions are characterized independently from one another. We then use an original technique to keep as few relevant dimensions as possible to describe each of these rules representing the *clusters*.

The rest of the paper is organized as follows: in section 2, we present existing *subspace clustering* methods and discuss their performances; we then describe our proposed algorithm called **SSC** in section 3; the results of our experiments, conducted on artificial as well as real databases, are then reported in section 4; finally, section 5 concludes the paper and suggests topics for future research.

2 Subspace Clustering

The *subspace clustering* problem has been recently introduced in [2]. Many other methods emerged then, among which two families can be distinguished according to their subspace search method:

1. *bottom-up* subspace search methods [2,6,8,9] that seek to find clusters in subspaces of increasing dimensionality, and produce as output a set of clusters that can overlap,
2. and *top-down* subspace search methods [1,7,12,13,15] that use *k-means like* methods with original techniques of local feature selection, and produce as output a partition of the dataset.

In [10], the authors have studied and compared these methods. They point out that every method requires input parameters difficult to set for the user, and that influence the results (density threshold, mean number of relevant dimensions of the clusters, minimal distance between clusters, etc.). Moreover, although a proposition was made to integrate discrete attributes in *bottom-up* approaches, all experiments were conducted on numerical databases only. Finally, let us note that no proposition was made for producing an interpretable output. This is however crucial because although dimensionality of clusters is reduced in the subspaces specific to them, it can still be too high so that a human user can easily understand it. Yet we will see that in many cases, it is possible to ignore some of these dimensions although keeping the same partition of the data.

The next section presents a new *subspace clustering* algorithm called **SSC** . It is *top-down like* and provides as output a set of clusters represented as rules that may overlap.

3 Algorithm SSC

Let us first denote by N the number of data points of the input database and M the number of dimensions on which they are defined. These dimensions can be continuous as well as discrete. We suppose values on continuous dimensions are normalized (so that all values belong to the same interval), and denote by $Categories_d$ the set of all possible categories on the discrete dimension d, and $Frequences_d$ the frequences of all these categories within the dataset.

3.1 Probabilistic model

One aim of this paper is to propose a probabilistic model that enables to produce an interpretable output. The basis of our model is the classical mixture of probability distributions $\theta = (\theta_1, ..., \theta_K)$ where each θ_k is the vector of parameters associated with the k^{th} cluster to be found, denoted by C_k (we set to K the total number of clusters). In order to produce an interpretable output, the use of rules (hyper-rectangles in subspaces of the original description space) is well suited because rules are easily understandable by humans. To integrate this constraint into the probabilistic model, we propose to add the hypothesis that data values follow independent distributions on each dimension. Thus, the new model is less expressive than the classical one that takes into account the possible correlations between dimensions. But it is adapted to the presentation of the partition as a set of rules because each dimension of each cluster is characterized independently from one another. Besides, the algorithm is thus faster than with the classical model because the new model needs less parameters ($O(M)$ instead of $O(M^2)$) and operations on matrices are avoided.

In our model, we suppose data follow gaussian distributions on continuous dimensions and multinomial distributions on discrete dimensions. So the model has the following parameters θ_k for each cluster C_k: π_k denotes its weight, μ_{kd} its mean and σ_{kd} its standard deviation on continuous dimensions d, and $Freqs_{kd}$ the frequences of each category on discrete dimensions d.

3.2 Maximum Likelihood Estimation

Given a set D of N data points $\overrightarrow{X_i}$, *Maximum Likelihood Estimation* is used to estimate the model parameters that best fit the data. To do this, the *EM algorithm* is an effective two-step process that seeks to optimize the *log-likelihood* of the model θ according to the dataset D, $LL(\theta|D) = \sum_i \log P(\overrightarrow{X_i}|\theta)$:

1. E-step (*Expectation*): find the class probability of each data point according to the current model parameters.
2. M-step (*Maximization*): update the model parameters according to the new class probabilities.

These two steps iterate until a stopping criterion is reached. Classicaly, it stops when $LL(\theta|D)$ increases less than a small positive constant δ from one iteration to another.

The E-step consists of computing the membership probability of each data point $\overrightarrow{X_i}$ to each cluster C_k with parameters θ_k. In our case, dimensions are assumed to be independent. So the membership probability of a data point to a cluster is the product of membership probabilities on each dimension. Besides, to avoid that a probability equal to zero on one dimension cancels the global probability, we use a very small positive constant ϵ.

$$P(\overrightarrow{X_i}|\theta_k) = \prod_{d=1}^{M} max(P(X_{id}|\theta_{kd}), \epsilon)$$

$$P(X_{id}|\theta_{kd}) = \begin{cases} \frac{1}{\sqrt{2\pi}\sigma_{kd}} e^{-\frac{1}{2}\left(\frac{X_{id}-\mu_{kd}}{\sigma_{kd}}\right)^2} & \text{if } d \text{ continuous} \\ Freqs_{kd}(X_{id}) & \text{if } d \text{ discrete} \end{cases}$$

$$P(\overrightarrow{X_i}|\theta) = \sum_{k=1}^{K} \pi_k \times P(\overrightarrow{X_i}|\theta_k) \text{ and } P(\theta_k|\overrightarrow{X_i}) = \frac{\pi_k \times P(\overrightarrow{X_i}|\theta_k)}{P(\overrightarrow{X_i}|\theta)}$$

Then the M-step consists of updating the model parameters according to the new class probabilities as follows:

$$\pi_k = \frac{1}{N} \sum_i P(\theta_k|\overrightarrow{X_i})$$

$$\mu_{kd} = \frac{\sum_i X_{id} \times P(\theta_k|\overrightarrow{X_i})}{\sum_i P(\theta_k|\overrightarrow{X_i})} \text{ and } \sigma_{kd} = \sqrt{\frac{\sum_i P(\theta_k|\overrightarrow{X_i}) \times (X_{id}-\mu_{kd})^2}{\sum_i P(\theta_k|\overrightarrow{X_i})}}$$

$$Freqs_{kd}(cat) = \frac{\sum_{\{i|X_{id}=cat\}} P(\theta_k|\overrightarrow{X_i})}{\sum_i P(\theta_k|\overrightarrow{X_i})} \; \forall \; cat \in Categories_d$$

It is well known that with the classical stopping criterion, convergence can be slow with *EM*. In order to make our algorithm faster, we propose to add the following *k-means like* stopping criterion: stop whenever the membership of each data point to their most probable cluster does not change. To do this, we introduce a new view on each cluster C_k, corresponding to the set S_k, of size N_k, of data points belonging to it: $S_k = \{\overrightarrow{X_i}|Argmax_{j=1}^{K} P(\overrightarrow{X_i}|\theta_j) = k\}$.

It is also well known that the *EM algorithm* results are very sensitive to the choice of the initial solution. So we run the algorithm many times with random initial solutions and finally keep the model optimizing the *log-likelihood* $LL(\theta|D)$.

At this stage, our algorithm needs one information from the user: the number of clusters to be found. This last parameter of the system can be found automatically with the widely used *BIC* criterion [14]:

$$BIC = -2 \times LL(\theta|D) + m_M \log N$$

with m_M the number of independent parameters of the model. BIC criterion must be minimized to optimize the likelihood of the model to the data. So, starting from $K = 2$, the algorithm with fixed K is run and BIC is computed. Then K is incremented, and iterations stop when BIC increases.

3.3 Output presentation

To make the results as comprehensible as possible, we now introduce a third view on each cluster corresponding to its description as a rule defined with as few dimensions as possible.

Relevant dimensions detection In order to select the relevant dimensions of the clusters, we compare on each dimension the likelihood of our model with that of a uniform model. Thus, if the likelihood of the uniform model is greater than the one of our model on one dimension, this dimension is considered to be irrelevant for the cluster. Let us first define the likelihood of a model θ' on a cluster C_k and a dimension d:

$$LL(\theta'|C_k, d) = \sum_{\overrightarrow{X_i} \in S_k} \log P(X_{id}|\theta')$$

In the case of a uniform model θ_{U_c} on continuous dimensions, as we suppose the database is normalized, we set $P(X_{id}|\theta_{U_c}) = 1$, and so $LL(\theta_{U_c}|C_k, d) = 0$. Thus, a continuous dimension d is considered to be relevant for a cluster C_k if

$$LL(\theta_{kd}|C_k, d) > 0$$

In the case of discrete dimensions, let θ_{U_d} be the uniform distribution. Then we set $P(X_{id}|\theta_{U_d}) = 1/|Categories_d|$. So $LL(\theta_{U_d}|C_k, d) = -N_k \times \log |Categories_d|$. For our model on discrete dimensions,

$$LL(\theta_{kd}|C_k, d) = \sum_{\overrightarrow{X_i} \in S_k} \log Freks_{kd}(X_{id})$$

As $LL(\theta_{kd}|C_k, d)$ is always greater than $LL(\theta_{U_d}|C_k, d)$ and both are negative, we need to introduce a constant $0 < \alpha < 1$ and set that d is relevant for the cluster if

$$LL(\theta_{kd}|C_k, d) > \alpha \times LL(\theta_{U_d}|C_k, d)$$

Dimension pruning Although we have already selected a subset of dimensions relevant for each cluster, it is still possible to prune some and simplify the clusters representation while keeping the same partition of the data.

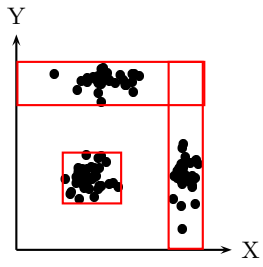


Fig. 1. Example of minimal description.

See figure 1 as an example. In that case, the cluster on the right is dense on both dimensions X and Y. So its true description subspace is $X \times Y$. However,

we do not need to consider Y to distinguish it from the other clusters: define it by high values on X is sufficient. The same reasoning holds for the cluster on the top.

To do this dimension pruning, we first create the rule R_k associated with the current cluster C_k. We now only consider the set of dimensions considered as relevant according to the previous selection. On continuous dimensions, we associate with the rule the smallest interval containing all the coordinates of the data points belonging to S_k. For discrete dimensions, we chose to associate with the rule the most probable category.

We then associate a weight W_{kd} with each dimension d of the rule R_k. For continuous dimensions, it is the ratio between local and global standard deviation according to μ_{kd}. And for discrete dimensions, it is the relative frequence of the most probable category.

$$W_{kd} = \begin{cases} 1 - \frac{\sigma_{kd}^2}{\sigma_d^2} \text{ , with } \sigma_d^2 = \frac{\sum_i (X_{id} - \mu_{kd})^2}{N} & \text{if } d \text{ continuous} \\ \frac{Freqs_{kd}(cat) - Frequences_d(cat)}{1 - Frequences_d(cat)} & \text{if } d \text{ discrete} \\ \text{with } cat = Argmax_{\{c \in Categories_d\}} Freqs_{kd}(c) & \end{cases}$$

We then compute the support of the rule (the set of data points comprised in the rule). This step is necessary since it is possible that some data points belong to the rule but not to the cluster. And finally, for all relevant dimensions presented in ascending order of their weights, delete the dimension from the rule if the deletion does not modify its support.

4 Experiments

Experiments were conducted on artificial as well as real databases. The first ones are used to observe the robustness of our algorithm faced with different types of databases. In order to compare our method with existing ones, we conducted these experiments on numerical-only databases. Then real databases are used to show the effectiveness of the method on real-life data (that may contain discrete attributes).

4.1 Artificial databases

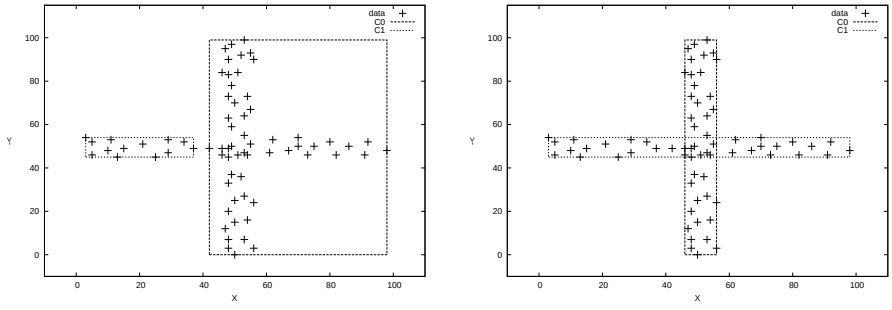
Artificial databases are generated according to the following parameters: N the number of data points in the database, M the number of (continuous) dimensions on which they are defined, K the number of clusters, MC the mean dimensionality of the subspaces on which the clusters are defined, SD_m and SD_M the minimum and maximum standard deviation of the coordinates of the data points belonging to a same cluster, from its centroid and on its specific dimensions.

K random data points are chosen on the M-dimensional description space and used as seeds of the K clusters $(C_1, ..., C_K)$ to be generated. Let us denote them by $(\overrightarrow{O_1}, ..., \overrightarrow{O_K})$. With each cluster is associated a subset of the N data

points and a subset (of size close to MC) of the M dimensions that will define its specific subspace. Then the coordinates of the data points belonging to a cluster C_k are generated according to a normal distribution with mean O_{kd} and standard deviation $sd_{kd} \in [SD_m..SD_M]$ on its specific dimensions d. They are generated uniformly between 0 and 100 on the other dimensions.

Our method is *top-down like*. Among the most recent ones, LAC [7] is an effective method that, as ours, only needs one user parameter: the number of clusters to be found (if we do not use BIC). So we propose to compare our method with LAC and provide to both algorithms the number of clusters to be found. LAC is based on *k-means* and associates with each centroid a vector of weights on each dimension. At each step and for each cluster, these weights on each dimension are updated according to the dispersion of the data points of the cluster on the dimension (the greater the dispersion, the less the weight).

Figure 2 shows the result of LAC and **SSC** on an artificial database. On this example, we can observe a classical limitation of *k-means like* methods over *EM like* methods: the first ones do not accept that data points belong to multiple clusters whereas the second ones give to each data point a membership probability to each cluster. Thus, contrary to *EM like* methods, *k-means like* methods are not able to capture concepts like the one appearing in figure 2 (one cluster is defined on one dimension and takes random values on another, and conversely for the other one) because of the intersection between clusters.



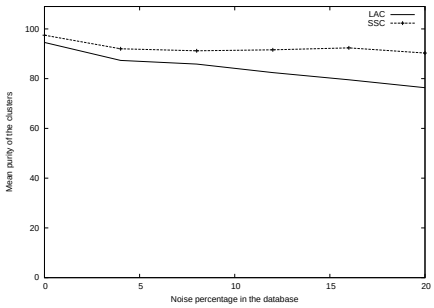
(a) LAC (b) SSC

Fig. 2. LAC versus SSC.

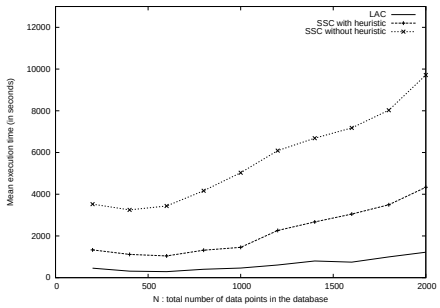
Experiments conducted on artificial databases with different generation parameters pointed out the robustness of our method. In particular, we observe that it is resistant to noise (see figure 3(a)). Accuracy of the partition is measured by the average purity of the clusters (the purity of a cluster is the maximum percentage of data points belonging to the same initial concept). With 20% of noise in the database, the average purity of the clusters is 90 for **SSC** while only 76 for LAC.

Our method is also robust to missing values. When summing over all data values on one dimension, the only thing to do is to ignore the missing values.

Concerning the execution time of our algorithm, experiments pointed out that the acceleration heuristic we proposed in section 3.2 is effective: for results of the same quality, the computing times of **SSC** with the heuristic are nearer to that of LAC (*k-means like* methods are well known for their efficiency) than to that of **SSC** without the heuristic (see figure 3(b)).



(a) Resistance to noise varying between 0 and 20%.



(b) Execution time according to $200 < N < 2000$.

Fig. 3. Artificial tests with $N = 600$, $M = 30$, $K = 5$, $MC = 3$, $SD_m = 2$, $SD_M = 5$.

Let us finally note that the results of our method are still robust even if data were generated by uniform distributions inside given intervals on the specific dimensions of the clusters, instead of normal distributions.

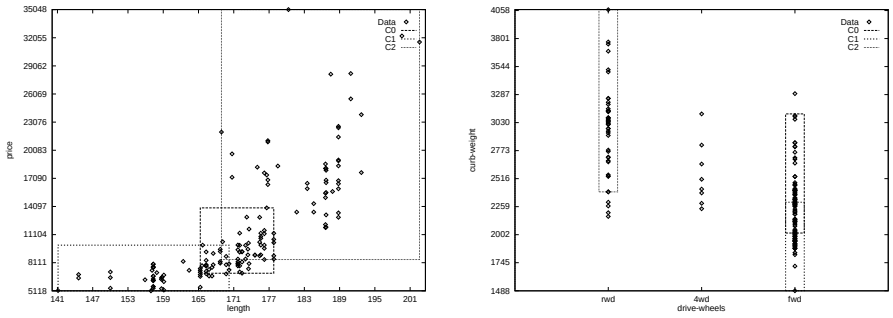
4.2 Real databases

Experiments were also conducted on real databases. Among them, the *Automobile* database coming from UCI repository [4] contains the description of 205 cars defined by a mix of 16 continuous and 10 discrete attributes. On this database, the three clusters found by **SSC** are characterized with a mean of only four dimensions. It thus points out that our method is effective in reducing the dimensionality, and thus giving an interpretable description of the clusters found.

Besides, this reduced description also allows us to compute with few cost a weight associated with each couple of relevant dimensions corresponding to the visualization power of this couple (remind W_{ki} is the weight, for the cluster C_k, of the dimension i):

$$V_{ij} = \sum_{k=1}^{K} max(W_{ki}, W_{kj})$$

The graphical visualizations corresponding to the two more visual couples of dimensions in the case of the *Automobile* database are provided figure 4. It thus visually shows that the price of cars increases a lot when their length exceeds 170 (figure 4(a)), that the cars with *rear-wheel drive* (*rwd*) have an average higher *curb-weight* than cars with *front-wheel drive* and *4-wheel drive* (figure 4(b)), and that the majority of the most expensive cars are *rear-wheel drive* (correspondance between both figures concerning cluster C_2).



(a) Projections on *length* and *price*.

(b) Projections on *drive-wheels* and *curb-weight*.

Fig. 4. Results of **SSC** on the *Automobile* database for $K = 3$.

5 Conclusion

We have presented in this paper a new *subspace clustering* method based on the use of a probabilistic model with the specific assumption that data were following independent distributions on each dimension. This idea has already been studied in [11]. But the method described by the authors differs from ours on some points. First, instead of using a mixture of gaussians on continuous dimensions, they use a mixture of uniform density M-dimensional hyper-rectangles supplemented with gaussian "tails", depending on a parameter σ that decreases during execution. Thus, their method is not adapted for incremental learning, whereas **SSC** can update its model when new data points arise. Moreover, we effectively integrated the problem of handling discrete dimensions whereas it was just mentioned as potential improvements in [11]. We have also proposed an original technique of dimension selection allowing us to provide as output an interpretable and visual representation of the clusters found.

Besides, we have proposed an original heuristic to speed up our algorithm. To continue our investigation in that direction, it seems interesting to take into account the work of [5] that is about the acceleration of the *EM algorithm* in the general case. Another way should be to consider only relevant dimensions during the iteration process.

Finally, we think it can be interesting to adapt our method for supervised or semi-supervised learning. And it should also be interesting to study the effectiveness of our method in a feature selection task.

References

1. Charu C. Aggarwal, Joel L. Wolf, Philip S. Yu, Cecilia Procopiuc, and Jong Soo Park. Fast algorithms for projected clustering. In *ACM SIGMOD Int. Conf. on Management of Data*, pages 61–72, 1999.
2. Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *ACM SIGMOD Int. Conf. on Management of Data*, pages 94–105, Seattle, Washington, 1998.
3. Pavel Berkhin. Survey of clustering data mining techniques. Technical report, Accrue Software, San Jose, California, 2002.
4. C.L. Blake and C.J. Merz. UCI repository of machine learning databases [http://www.ics.uci.edu/∼mlearn/MLRepository.html], 1998.
5. P. Bradley, U. Fayyad, and C. Reina. Scaling EM (Expectation-Maximization) clustering to large databases. Technical report, Microsoft Research, Aug. 1998.
6. Chun Hung Cheng, Ada Wai-Chee Fu, and Yi Zhang. Entropy-based subspace clustering for mining numerical data. In *Knowledge Discovery and Data Mining*, pages 84–93, 1999.
7. Carlotta Domeniconi, Dimitris Papadopoulos, Dimitrios Gunopolos, and Sheng Ma. Subspace clustering of high dimensional data. In *SIAM Int. Conf. on Data Mining*, 2004.
8. Karin Kailing, Hans-Peter Kriegel, and Peer Kröger. Density-connected subspace clustering for high-dimensional data. In *SIAM Int. Conf. on Data Mining*, pages 246–257, 2004.
9. Harsha Nagesh, Sanjay Goil, and Alok Choudhary. Mafia: Efficient and scalable subspace clustering for very large data sets. Technical report, Northwestern University, 1999.
10. Lance Parsons, Ehtesham Haque, and Huan Liu. Evaluating subspace clustering algorithms. In *Workshop on Clustering High Dimensional Data and its Applications, SIAM Int. Conf. on Data Mining*, pages 48–56, 2004.
11. Dan Pelleg and Andrew Moore. Mixtures of rectangles: Interpretable soft clustering. In Carla Brodley and Andrea Danyluk, editors, *18th Int. Conf. on Machine Learning*, pages 401–408. Morgan Kaufmann, San Francisco, California, 2001.
12. Ioannis A. Sarafis, Phil W. Trinder, and Ali M. S. Zalzala. Towards effective subspace clustering with an evolutionary algorithm. In *IEEE Congress on Evolutionary Computation*, Canberra, Australia, Dec. 2003.
13. Kyoung-Gu Woo and Jeong-Hoon Lee. *FINDIT: a fast and intelligent subspace clustering algorithm using dimension voting*. PhD thesis, Korea Advanced Institute of Science and Technology, Department of Electrical Engineering and Computer Science, 2002.
14. L. Ye and M.E. Spetsakis. Clustering on unobserved data using mixture of gaussians. Technical report, York University, Toronto, Canada, Oct. 2003.
15. Kevin Y. Yip, David W. Cheung, and Michael K. Ng. A highly-usable projected clustering algorithm for gene expression profiles. In *3rd ACM SIGKDD Workshop on Data Mining in Bioinformatics*, pages 41–48, 2003.