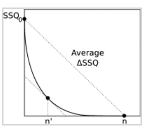

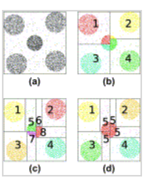

Information Sciences

Volume 262, 20 March 2014, Pages 32–45

Analysing microarray expression data through effective clustering

E. Masciari[a], , G.M. Mazzeo[a], , C. Zaniolo[b],

Show more

DOI: 10.1016/j.ins.2013.12.003

Abstract

The recent advances in genomic technologies and the availability of large-scale microarray datasets call for the development of advanced data analysis techniques, such as data mining and statistical analysis to cite a few. Among the mining techniques proposed so far, cluster analysis has become a standard method for the analysis of microarray expression data. It can be used both for initial screening of patients and for extraction of disease molecular signatures. Moreover, clustering can be profitably exploited to characterize genes of unknown function and uncover patterns that can be interpreted as indications of the status of cellular processes. Finally, clustering biological data would be useful not only for exploring the data but also for discovering implicit links between the objects. To this end, several clustering approaches have been proposed in order to obtain a good trade-off between accuracy and efficiency of the clustering process. In particular, great attention has been devoted to hierarchical clustering algorithms for their accuracy in unsupervised identification and stratification of groups of similar genes or patients, while, partition based approaches are exploited when fast computations are required. Indeed, it is well known that no existing clustering algorithm completely satisfies both accuracy and efficiency requirements, thus a good clustering algorithm has to be evaluated with respect to some external criteria that are independent from the metric being used to compute clusters. In this paper, we propose a clustering algorithm called M-CLUBS (for Microarray data CLustering Using Binary Splitting) exhibiting higher accuracy than the hierarchical ones proposed so far while allowing a faster computation with respect to partition based approaches. Indeed, M-CLUBS is faster and more accurate than other algorithms, including k-means and its recently proposed refinements, as we will show in the experimental section. The algorithm consists of a divisive phase and an agglomerative phase; during these two phases, the samples are repartitioned using a least quadratic distance criterion possessing unique analytical properties that we exploit to achieve a very fast computation. M-CLUBS derives good clusters without requiring input from users, and it is robust and impervious to noise, while providing better speed and accuracy than methods, such as BIRCH, that are endowed with the same critical properties. Due to the structural feature of microarray data (they are represented as arrays of numeric values), M-CLUBS is suitable for analyzing them since it is designed to perform well for Euclidean distances. In order to stronger the obtained results we interpreted the obtained clusters by a domain expert and the evaluation by quality measures specifically tailored for biological validity assessment.

Keywords

Bioinformatics; Clustering; Biological data analysis

1. Introduction

Nowadays, microarray experiments allow the exploration of huge amounts of gene expressions using a single chip. Moreover, the relatively moderate cost for a chip and the small sample preparation times, enable the analysis of a large number of different experimental conditions, such as points of time-series experiments or disease progression in a cohort of patients [33].

This huge amount of data poses many challenges to the bioinformatics community such as finding the behavior of set of related genes in different conditions. This goal is often achieved by means of cluster analysis, i.e. the identification of similar patterns in different conditions [25]. Indeed, the ability to gather genome-wide expression data has far outstripped the ability of human brains to process the raw data, thus cluster analysis can help scientists to distill the data down to a more comprehensible level by subdividing the genes into a smaller number of categories and then analyzing those [7], [9] and [15].

Further motivation for the exploitation of cluster analysis for biological data lies in the fact that similar patterns found by clustering may correspond to co-regulation of genes [21]. Moreover, cluster analysis represents a fundamental and widely used method of knowledge discovery [26], due to the valuable information it can provide. In particular, the use of cluster analysis has become a standard method in literature for the analysis of microarray expression data used both for initial screening of patients as well as for extraction of molecular signatures of disease [24] or feature selection [5] and [30]. By cluster analysis, microarray data researcher can focus on finding group of genes that exhibit a similar and coherent evolutionary patterns in a set of patients or time-points. For instance Bayesian approaches have been largely used for data analysis, but their limited scalability and efficiency prevent their use in large scale microarray datasets [27], [28] and [39]. Analogously, a large number of existing algorithms has been applied to microarray data starting from well-known approaches; among those we mention here partition-based clustering (e.g. *k-means* [36]) and its variants (e.g. *fuzzy c-means* [14]), density based clustering (e.g. *DBScan* [17]), hierarchical methods (e.g. *BIRCH* [47], R/BHC [40]), and grid-based methods (e.g. *STING* [44] and [45]). In particular, agglomerative hierarchical clustering has been used to partition set of patients into smaller groups characterized by exploiting information on set of genes exhibiting similar evolution with respect to a set of similar conditions (e.g. clinical conditions, time evolution or drug responses) [32].

Nevertheless, the logical and algorithmic complexities of this many-facet problem make this research activity quite intriguing. Indeed, in spite of the new progress achieved in recent years (e.g., agglomerative clustering [34], biclustering [1], genetic algorithm based clustering [35], non-metric clustering [19]), significant progress should be expected in the future. In particular, it is well known that no clustering algorithm completely satisfies both accuracy and efficiency requirements, thus a good clustering algorithm has to be evaluated with respect to some external criteria that are independent from the metric being used to compute clusters. As an example, bootstrapping techniques have often been used to calculate the significance of the obtained dendrogram [29].

In this paper, we propose M-CLUBS, a novel algorithm that exhibits quite good performances, in term of *speed, repeatability, accuracy* and *robustness to noise*. M-CLUBS performances have been evaluated using widely accepted clustering validity metric that are method independent thus quite reliable. M-CLUBS excellent performances arise from some key feature of our algorithm, in particular:

- M-CLUBS is not tied to a fixed grid differently from grid-based methods (e.g. *STING* [44]),
- it can backtrack on previously wrong calculation since it performs first a top-down splitting of data and then (eventually) it performs a bottom-up refinement of the obtained results,
- it performs also well on non-globular clusters (i.e. clusters that are not spherical in shape) differently from *k-means* [36] and *BIRCH* [47].

In the following, after a presentation of our method, we show the M-CLUBS properties and finally as proof-of-principle we discuss the performance of our algorithm using some publicly available dataset.

2. Approach

In this paper we propose a new hierarchical algorithm called M-CLUBS (for Microarray data CLustering Using Binary Splitting) whose *speed* performances are better than *k*-means *and* whose *accuracy* overcomes previous hierarchical algorithms while operating in a *completely unsupervised* fashion. The first phase of the algorithm is divisive, as the original data set is split recursively into mini-clusters through successive binary splits: the algorithm's second phase is agglomerative since these mini-clusters are recombined into the final result. Due to its features our algorithm can be used also for refining other approaches performances. As an example it can be used to overcome *k*-means initial assignment problem since its low complexity will not affect the overall complexity while the accuracy of our results will guarantee an excellent initial assignment of cluster centroids. Further, our approach induces during execution a *dynamic hierarchical* grid that will better fit the dataset with respect to classical grid approaches that exploit a fixed grid instead. Finally, the algorithm exploits the analytical properties of the Sum of Squares (SSQ in the following) function to minimize the cost of merge and split operations, and indeed the approach results really fast. One may argue that many different measures could be used for cluster computation but the accuracy of SSQ is as good as other cluster distance measures, such as Single Link, Complete Link, Average (see Section 4) for real case scenarios *and* its computation can be made faster than other measures.

Main Difference of M-CLUBS with respect to other approaches. M-CLUBS works in a completely unsupervised way and overcomes the main limitations that beset other algorithms. In particular, we have that (1) M-CLUBS is not tied to a fixed grid, (2) it can backtrack on previously wrong calculation, and (3) it performs also well on non-globular clusters where clusters are not spherical in shape, this feature will be intuitively understood after the partitioning and recombination strategy will be detailed in next section (BIRCH does not perform as well, because it uses the notion of radius or diameter to control the boundary of a cluster, and the same drawback also affects *k*-means like algorithms). Moreover we have that (4) M-CLUBS can detect the natural clusters present in data, while in Birch each node in the auxiliary tree exploited (called CF tree) can hold only a limited number of entries due to its size thus a CF tree node does not always correspond to what a user may consider a natural cluster. Finally, (5) density based algorithms like DBSCAN are very sensitive to clustering parameters like Minimum Neighborhood Points and they fail to identify clusters if density varies and if the data set is too sparse and different sampling affects density measures, however we compared M-CLUBS against OPTICS that allows to detect clusters with different densities instead. As will be clear by experimental evaluation, M-CLUBS does not suffer these limitations due to the unique feature of SSQ and the two-phase algorithm.

Another relevant parameter to take into account is the computational cost of the algorithms. In general hierarchical algorithms are slower than partition based algorithms like *k-means*. Indeed, *k-means* is really fast but the accuracy of the results could be not satisfactory, moreover *k-means* depends on the choice of the number of cluster *k* and the initial assignment of the cluster centers. In particular, a wrong choice of the initial cluster centers lead to an incorrect clustering. Indeed, M-CLUBS offers a good accuracy while performing a really fast computation.

3. Methods

We first recall some basic notions exploited by our algorithm then we discuss our binary partitioning strategy and the cluster quality measures we used to evaluate the obtained results. Throughout the paper, for each dataset a d -dimensional data distribution D is assumed. D will be treated as a multi-dimensional array of integers with volume n^d (without loss of generality, we assume that all dimensions of D have the same size). The number of non-zero elements of D will be denoted as N . A *range* ρ_i on the i th dimension of D is an interval $[1 \dots u]$, such that $1 \leqslant 1 \leqslant u \leqslant n$. Boundaries l and u of ρ_i are denoted by $lb(\rho_i)$ (*lower bound*) and $ub(\rho_i)$ (*upper bound*), respectively. The size of ρ_i will be denoted as $size(\rho_i) = ub(\rho_i) - lb(\rho_i) + 1$. A *block* b (of D) is a d -tuple $\langle \rho_1, \dots, \rho_d \rangle$ where ρ_i is a range on the dimension i , for each $1 \leqslant i \leqslant d$. Informally, a block represents a “hyper-rectangular” region of D . A block b of D with all zero elements is said to be a *null block* . The volume of a block $b = \langle \rho_1, \dots, \rho_d \rangle$ is given by $size(\rho_1) \times \dots \times size(\rho_d)$ and will be denoted as $vol(b)$. Given a point in the multidimensional space $\mathbf{x} = \langle x_1, \dots, x_d \rangle$, we say that $\mathbf{x}$ belongs to the block b (written $\mathbf{x} \in b$) if $lb(\rho_i) \leqslant x_i \leqslant ub(\rho_i)$ for each $i \in [1 \dots d]$.

Given a block $b = \langle \rho_1, \dots, \rho_d \rangle$, let x be a coordinate on the i th dimension of b such that $lb(\rho_i) \leqslant x < ub(\rho_i)$. Coordinate x divides the range ρ_i of b into and , thus partitioning b into and . The pair $\langle b^{low}, b^{high} \rangle$ is said to be the *binary split* of *b* along the dimension *i* at the position *x*; dimension *i* and coordinate *x* are said to be the *splitting dimension* and the *splitting position*, respectively.

Informally, a binary partition can be obtained by performing a binary split on D (thus generating the two sub-blocks D^{low} and D^{high}), and then recursively partitioning these two sub-blocks with the same binary hierarchical scheme.

Definition 1.

Given a d -dimensional data distribution D with volume n^d, a *binary partition BP* of D is a binary tree such that the root of BP is the block $\langle [1 \dots n], \dots, [1 \dots n] \rangle$ and for each internal node p of BP the pair of children of p is a binary-split of p . □

Given a dataset DS cluster analysis aims at producing a clustering $C = \{C_1, \dots, C_n\}$ that is a subset of the set of all subsets of DS such that C contains disjoint (non-overlapping) subsets, covering the whole object set (we refer in this paper exclusively to hard clustering problem, where every data point belongs to one and only one cluster). Consequently, every point $x \in DS$ is contained in exactly one and only one set C_i. These sets C_i are called clusters.

Definition 2.

Let $\mathrm{C_s}$ be a cluster (set) of *N d* -dimensional points. Let $\mathbf{S}=(\mathrm{S}_1,\ldots,\mathrm{S}_\mathrm{d})=\sum_{\mathbf{p}\in \mathrm{C_s}}\mathbf{p}$ be the vector representing the sum of points in $\mathrm{C_s}$. The center of $\mathrm{C_s}$ is . Let $\mathbf{Q}=(\mathrm{Q}_1,\ldots,\mathrm{Q}_\mathrm{d})$, where , be the vector whose *i* th coordinate is the sum of the squared *i* th coordinates of the points in *S* . The SSQ (Sum of Squares) of $\mathrm{C_s}$ is defined as:

Turn on

we recall that *N* is the number of points in *C* and

thus we obtain by substituting:

finally by definition of $\mathrm{Q_i}$ and $\mathrm{S_i}$ we obtain:

(1)

From the latter, it is clear that, in order to quickly compute the SSQ of a cluster, we need only to store , and *N*. In the next section we will show how these information can be used effectively and efficiently to optimize the divisive and agglomerative steps of the M-CLUBS algorithm.

3.1. Our clustering approach

In order to obtain a good trade-off between accuracy and efficiency we exploit in this paper a new fast hierarchical approach. Among hierarchical algorithms, bottom-up approaches tend to be more accurate but have a higher computational cost than the top-down approaches [26]. The higher cost is due to the higher number of candidate clusters to be taken into account. To overcome this limitation, in our approach, the agglomerative step is only used on mini-clusters generated by a first divisive process, this results in a remarkable efficiency increase. Top-down partitioning exploiting greedy algorithms has been widely used in the multidimensional data compression due to its efficiency. Here we use a similar divisive approach to minimize the SSQ among the data belonging to clusters, we recall again that in literature many measures have been proposed, e.g. EES (Error of Estimates) [42] that works in a similar way as SSQ but we chose SSQ since it offers a really fast computation while maintaining an high accuracy in cluster model evaluation. Thus, our clustering algorithm consists of two steps, where in the first step we use binary hierarchical partitioning to produce a set of mini-clusters and in the second step, we pairwise merge the mini-clusters so obtained in a bottom-up fashion. In both steps the clusters are defined by a hierarchical partition of the multi-dimensional space. The partition can be compactly represented by a binary tree, where: (1) each node is associated with a range of the multi-dimensional domain; (2) the root is associated with the whole data domain; (3) for each inner node *n*, its children are associated with a pair of ranges representing a (rectangular) partition of *n*.

Each node also maintains summary information about points inside its range, to expedite the clustering computation. The top-down splitting works as follows. As auxiliary structure, we maintain a priority queue of clusters whose elements are ordered on the basis of the SSQ of each cluster. At each iteration, the algorithm performs the following two steps: (A) select the cluster $\mathrm{C_s}$ that exhibits the highest SSQ (i.e. the one on top of the priority queue) and then (B) partition this $\mathrm{C_s}$ in such a way that the SSQ reduction, denoted ΔSSQ, is maximized. For step B, we use formula (3) (reported next) to compute $\Delta SSQ(\mathrm{i},\mathrm{j})$ for each dimension *i* and for each cutting position *j* ; then we choose the position *j* that guarantees the maximum ΔSSQ. This computation can be done very efficiently since we pre-compute *Q* and *S* , and therefore we need a single scan of the data. We repeat these two steps, A and B above, while ΔSSQ is greater than the average SSQ. We recall that the partition (i.e., the cluster tree) is built by exploiting a greedy strategy. To this end, the tree is constructed top-down, by means of leaf-node splitting. At each step, the leaf with the largest SSQ is chosen, and it is split as to maximize the SSQ reduction, denoted ΔSSQ. Being SSQ a measure of range skewness, we perform splits as long as ΔSSQ remains "significant". After the early splits that yield large SSQ reductions, the values of ΔSSQ become smaller and smaller, until after n splits both SSQ and ΔSSQ become 0 (since each point has become its own cluster). Thus, the average SSQ reduction per split is SSQ_0/n, and we will compare this value against the current ΔSSQ to decide when we should stop splitting. The rationale for this criterion is clearly illustrated by Fig. 1, where the typical ΔSSQ slope is displayed against the average SSQ: there is no gain in splitting beyond the turning point (marked with a solid circle) since the SSQ reduction is less than the average ΔSSQ and thus imputable to random distributions rather than cluster-like ones.

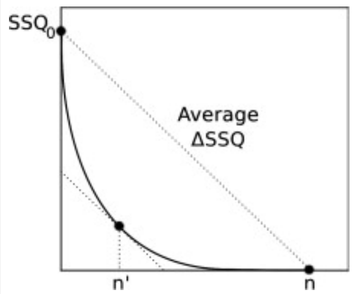

Fig. 1.

Average SSQ and ΔSSQ example plots.

Figure options

The splitting process just described is tied to the grid partitioning and thus may cause a non-optimal splitting of some clusters. The successive phase overcomes this limitation since the merging is performed considering all the possible pairs of adjacent mini-clusters, and recombining those that offer best SSQ reduction. This agglomerative process offers significant advantages. The first advantage is that it merges clusters in different grid partitions. This backtracking step overcomes non-optimal splits obtained in the first phase as it is easy to see in Fig. 3(b and c). The second critical advantage is that the computational complexity of this bottom-up step is very low since the number of merging steps is related to the number of clusters that is very low compared to usual dataset sizes. The final advantage is that this phase also halts automatically, producing an algorithm that does not require any seeding or other parameters from the user a really nice feature that is not shared by all clustering algorithms.

Remarks about M-CLUBS unsupervised features. Every clustering algorithm so far proposed, despite the unsupervised nature of clustering, requires some user interaction. *k*-means requires an initial centroids assignment, thus the expected number of clusters that indeed should be a priori unknown, hierarchical clustering algorithms require a termination condition, e.g. the desired number of clusters or the diameter of each cluster, but these information cannot be set with a perfect confidence. Similar problems occur for grid cells definition or exploitation of density information. Indeed, M-CLUBS overcome these limitations by working in a fully unsupervised way that greatly decrease the complexity of the algorithm while keeping a high accuracy. Moreover, traditional approaches can never undo what was done in previous computation steps and they are really sensitive to cluster distance measures. Finally, Birch performs an agglomerative step on micro-clusters but it exploits at the later macro-clustering stage different clustering methods such as iterative partitioning thus mixing different strategies.

3.2. M-CLUBS: a new clustering algorithm for microarray data

Fig. 2 provides a more formal description of the M-CLUBS algorithm. Note that in the declaration steps, *Vars* denotes the variables used in the corresponding subroutines. We use the notation $\mathrm{x} \cdot \mathrm{y}$ to denote step y in subroutine invoked at step x of the main. We use the *initializeTree* (Step 1) to load the dataset into the root of the auxiliary tree structure *BT* exploited for partitioning. Once the tree structure has been initialized the *topdownsplitting* starts (Step 2). In particular, the root of *BT* is added to a priority queue whose ordering criterion is based on the SSQ values of clusters stored in the queue. The initial cluster assignment performed by *initializeClusters* is composed by the root r of *BT* and the initial SSQ is the one computed on r (Steps 2.1–2.3). The function *computeAverageDeltaSSQ* averages the actual SSQ for all the points in the cluster

(Step 2.4). The function *computeWeightedDeltaSSQ* is (iteratively) applied to the cluster C_s that is currently on top of the priority queue (Step 2.7). The $weighted\Delta_{SSQ}$ is computed as the average gain of SSQ obtained by splitting C_s as explained above for Δ_{SSQ}, i.e. we pre-compute the marginal sums (S and Q) for a given splitting point (with respect to the coordinates ordering) and reassigning the splitting point based on these partial sums. In order to improve the effectiveness of splits the value of Δ_{SSQ} is raised to a power $p, p<1$, thus obtaining $weighted\Delta_{SSQ}$ value. If $weighted\Delta_{SSQ}$ is greater than *avgDeltaSSQ* computed by *computeAverageDeltaSSQ* then we proceed with the split (Step 2.9), otherwise we do not. We use values of p that are less than 1, since for $p \geqslant 1$ we would end up splitting clusters where the gain does not exceed the average Δ_{SSQ} associated with a random distribution. This would result in a large number of small clusters, where both intra-cluster and inter-cluster distances are small. We instead seek values of p that reduce the former while magnifying the latter. We determined that the best value is $p=0.8$ regardless the dataset feature, thus the user is not required to set any parameter. When no more top-down splits are possible, the *topDownSplitting* ends and we begin the *bottomUpMerging* (Step 3). In order to obtain more compact clusters, we select (by running *selectBestPair* at Step 3.2) the pair of clusters that, if merged, yields the least SSQ increase (that is assigned to *minInc* by function *computeSSQIncrease* run at Step 3.3). This merging step is repeated until *minInc* becomes larger than *avgDeltaSSQ* (Steps 3.4–3.9). Fig. 3 shows the algorithm in action. After three steps, the initial samples in Fig. 3(a) are partitioned according to the grid shown in Fig. 3(b). The algorithm takes seven more splitting steps producing the partition of Fig. 3(c). The merging phase produces the final five clusters that a human will instinctively recognize at a glance Fig. 3(d).

Input:
A dataset DS of n points
Output:
A set of clusters C.
Vars:
An auxiliary binary tree BT;
An initial cluster assignment C'.
Method: M-CLUBS

$BT :=$ `initializeTree`(DS);
$C' :=$ `topDownSplitting`(BT);
$C :=$ `bottomUpMerging`(C');
return C;

Function `topDownSplitting`$(BT) : C'$;
Vars:
A priority queue PQ;
A boolean $finished$;
A double Δ_{SSQ};
A double $avgDeltaSSQ$;
Method:

$PQ :=$ `add`$(BT.root())$;
$C' =$ `initializeClusters`;
$finished = false$;
$avg\Delta_{SSQ} =$ `computeAverageDeltaSSQ()`;
while $!finished$ **do begin**
  $C_s = PQ.get()$;
  $weighted\Delta_{SSQ} =$ `computeWeightedDeltaSSQ`(C_s);
  if $(weighted\Delta_{SSQ} > avg\Delta_{SSQ})$ **then**
    $C' :=$ `update`(C');
  else $finished = true$;
end while
return C';

Function `bottomUpMerging`$(C') : C$;
Vars:
A pair of cluster $Pair$;
A double $avgDeltaSSQ$;
A double $minInc$;
Method:

$C := C'$;
$Pair :=$ `selectBestPair`(C');
$minInc :=$ `computeSSQIncrease`$(Pair)$;
$avgDeltaSSQ =$ `computeAverageDeltaSSQ()`;
while $minInc < avgDeltaSSQ$ **do begin**
  $C :=$ `merge`$(Pair)$;
  $Pair :=$ `selectBestPair`(C);
  $minInc :=$ `computeSSQIncrease`$(Pair)$;
end while;
return C;

Fig. 2.

The M-CLUBS clustering algorithm.

Figure options

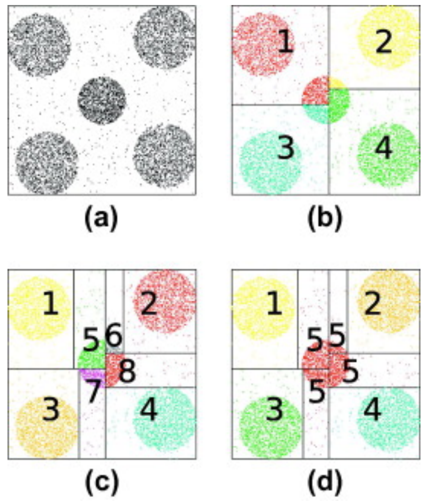

Fig. 3.

Execution steps of M-CLUBS.

Figure options

We point out that producing axis parallel cuts is not a limitation, we can still obtain, in our approach, nonparallel cuts however this will not improve the performances of the algorithm. Furthermore, also grid based approaches are tied to parallel cuts since they allow more efficient computation without paying any accuracy loss.

Due to its grid-based divisive agglomerative approach M-CLUBS produces cluster results of superior quality. Furthermore the algorithm can be executed with superior time and accuracy performances using a simple (and fast) formula that allows us to estimate the SSQ reduction produced by a split.

The above described approach is really efficient, in this respect we now discuss its computational complexity. We point out that algorithms exhibiting low computational complexity (while saving result accuracy) are particularly suited for biological data analysis due to the high size and high dimensionality of the available datasets. The following proposition is devoted to discuss our worst case complexity.

Proposition 1.

Algorithm M-CLUBS works in $O(n \cdot d \cdot l \cdot s)$ *where n is the number of points, d is the number of dimensions, l is the number of splitting positions for each dimension and s is the number of splits.*

To prove the proposition, we recall that, in order to perform splits, we have to compute the SSQ for each dimension and for each splitting point. Thus, each split has a complexity $O(n \cdot d \cdot l)$ and we perform s splits. The bottom-up step contributes to the overall complexity with a term $O(k^2)$ where k is the number of clusters, since for each cluster we have to consider only the possibly adjacent clusters for merging; but since $k \ll n$ (otherwise cluster assignment will be meaningless [26]) we can disregard this term leading to $O(n \cdot d \cdot l \cdot s)$ complexity for the worst case scenario.

4. Discussion

In order to show the characteristics of M-CLUBS we used two publicly available dataset on Gene Expression Omnibus Database: a dataset provided by [22], *Dataset 1* hereafter, and a dataset provided by [16], *Dataset 2* hereafter. Furthermore, we tested our algorithms on dataset *AD400-10-10* [46] and dataset *Yeast Sporulation* [11]. Analogously to [12] we compared several clustering algorithms in order to assess the validity of our approach in the biological data scenario. In particular, we compared our method with *BIRCH* [47], *K-means++* [4] (we refer to it as *KM++*), k^**-means* [10] (we refer to it as *SMART*), *OPTICS* [2] and *DIANA* [31]. For the k-means based algorithms we performed 20 runs (same as [4]) and we report the average values for these runs. Moreover, since our algorithm is hierarchical we compared it with respect to Single Link [41] (usually referred as *Nearest Neighbour Clustering*, we refer to it as *NN* in the following), Complete Link [13] (usually referred as *Farthest Neighbour Clustering*, we refer to it as *FN* in the following), Average approaches [23] (usually referred as *Unweighted Pair Group Method with Arithmetic Mean*, we refer to it as *UPGMA* in the following).

All the approaches mentioned above address the clustering problem from different viewpoints thus strengthening our evaluation. Finally, for the sake of completeness, we also ran several experiments using an algorithm designed for biological data as *SiMM-TS* [6] that confirmed our superior performances as will be shown below.

We started our analysis considering these datasets on which we used M-CLUBS and the other clustering algorithms for the sake of comparison. The obtained results are reported in Table 1 and Table 2.

Table 1.

Accuracy and time performances for Dataset1 and Dataset2.

Algorithm	Test datasets			
	$Dataset1$		$Dataset2$	
	SSQ	Time	SSQ	Time
M-CLUBS	**2.01E+8**	**2.513**	**1.77E+2**	**0.0784**
OPTICS	3.55E+8	5.271	1.79E+2	0.2456
BIRCH	2.67E+8	9.124	1.78E+2	0.3522
KM++	4.31E+8	2.913	1.82E+2	0.1154
SMART	4.65E+8	3.025	1.81E+2	0.1243
DIANA	3.96E+8	3.113	1.85E+2	0.1463
UPGMA	4.03E+8	6.412	1.91E+2	0.4331
NN	4.11E+8	6.635	1.86E+2	0.3992
FN	4.18E+8	6.935	1.88E+2	0.4021
SiMM-TS	2.99E+8	5.648	1.80E+2	0.4415

Values represent SSQ per dataset. Times are expressed in seconds.

Table options

Table 2.

Accuracy and time performances for AD400-10-10 and Yeastsporulation.

Algorithm	Test datasets			
	$AD400\text{-}10\text{-}10$		$Yeastsporulation$	
	SSQ	Time	SSQ	Time
M-CLUBS	**9.40E+4**	**0.831**	**2.41E+3**	**0.2451**
OPTICS	1.21E+5	1.025	3.98E+3	0.3125
BIRCH	1.43E+5	1.336	3.86E+3	0.4006
KM++	1.14E+5	0.992	3.77E+3	0.2998
SMART	1.32E+5	1.022	3.85E+3	0.3134
DIANA	1.03E+5	1.423	3.56E+3	0.3321
UPGMA	1.19E+5	1.551	3.68E+3	0.3779
NN	2.04E+5	1.352	3.80E+3	0.3881
FN	2.11E+5	1.398	3.88E+3	0.3967
SiMM-TS	9.87E+4	1.004	2.86E+3	0.3027

Values represent SSQ per dataset. Times are expressed in seconds.

Table options

The results obtained are quite convincing both for the accuracy and the execution times where M-CLUBS exhibits best performances (best results for each table are reported in bold). In particular our clustering method correctly detected the number of clusters in the data as stated in detail in next section. Indeed, M-

CLUBS showed a nice feature when clustering Dataset 1: the HN group contains two subgroups ER+ and ER-, M-CLUBS during the splitting step identified these two subgroups that have been collapsed in a single cluster after the merging step. To further asses, the validity of the approach we exploited several *method-independent* quality measure that are reported in the following.

4.1. Quality of clustering results

Here we will evaluate the quality of the results M-CLUBS produces and its reliability. The issue of finding method-independent measures for clustering results has been the source of much topical discussions, but over time sound measures have emerged that can be used reliably to compare the quality of the results produced by a wide range of clustering algorithms [8]. In particular the following three measures have sound theoretical and practical bases: *Variance Ratio* (its range is $[0,\infty)$ and larger values indicate better clustering quality), *Relative Margin* (its range is $[0,1)$ and lower values indicates a better clustering) and *Weakest Link* (its range is $[0,\infty)$ and lower values represent better clusterings).

The results obtained for the above mentioned quality measures are given in Table 3 and Table 4: they show that M-CLUBS outperforms other methods significantly, producing values for Relative Margin & Weakest Link (resp. Variance Ratio) that are significantly lower (larger) than those other methods, i.e. clusters of much better quality.

Table 3.

Clustering quality measures evaluation.

	#Clusters	Variance ratio	Relative margin	Weakest link
Dataset 1				
M-CLUBS	**3**	**75.41**	**0.098**	**0.817**
OPTICS	5	56.18	0.135	2.045
BIRCH	6	63.42	0.176	1.934
KM++	**3**	65.44	0.157	4.152
SMART	**3**	64.77	0.198	4.789
DIANA	4	66.16	0.121	1.921
UPGMA	6	63.56	0.197	2.442
NN	6	66.78	0.184	2.113
FN	6	67.16	0.178	2.241
SiMM-TS	4	69.83	0.115	1.443
Dataset 2				
M-CLUBS	**4**	**81.33**	**0.066**	**0.713**
OPTICS	**4**	67.18	0.153	1.876
BIRCH	**4**	70.41	0.182	1.943
KM++	**4**	68.67	0.201	3.412
SMART	**4**	69.97	0.225	3.725
DIANA	**4**	69.54	0.158	1.992
UPGMA	**4**	71.15	0.177	1.957
NN	**4**	70.93	0.184	1.964
FN	**4**	71.04	0.188	1.981
SiMM-TS	**4**	75.42	0.104	1.144

Table options

Table 4.

Clustering quality measures evaluation.

	#Clusters	Variance ratio	Relative margin	Weakest link
AD400-10-10				

M-CLUBS	**10**	**88.32**	**0.104**	**0.183**
OPTICS	9	76.42	0.166	0.913
BIRCH	9	78.44	0.181	1.036
KM++	**10**	79.31	0.194	1.231
SMART	**10**	78.21	0.206	1.312
DIANA	**10**	77.36	0.159	1.012
UPGMA	9	74,86	0.161	1.431
NN	9	76.62	0.183	1.532
FN	9	77.95	0.185	1.476
SiMM-TS	**10**	81.36	0.128	0.463
Yeast sporulation				
M-CLUBS	**7**	**83.45**	**0.153**	**0.147**
OPTICS	6	77.28	0.265	1.221
BIRCH	6	80.36	0.382	1.013
KM++	**7**	76.21	0.323	1.904
SMART	**7**	75.43	0.244	1.975
DIANA	8	78.42	0.297	1.146
UPGMA	6	77.03	0.342	1.442
NN	6	76.79	0.401	1.451
FN	6	76.31	0.414	1.433
SiMM-TS	**7**	81.43	0.195	0.348

Table options

These results show that M-CLUBS always finds the exact number of clusters and the quality of the found cluster is overwhelming with respect to the other methods.

4.2. Additional quality measures

SSQ is a natural and widely used norm of similarity, but a devil's advocate can point out that other clustering algorithms might not measure their effectiveness in terms of SSQ or even the compactness of each cluster around its centroid. Thus, we will attempt to measure the quality of the clusters produced by M-CLUBS using very different criteria inspired by the nearest subclass classifiers that were previously used in a similar role in [43] and [18].

A first relevant evaluation measure in this approach is the error rate of a k -Nearest Neighbor classifier defined by the clustering results. This value provide relevant information about the ability of the clustering method under evaluation to minimize the errors due to incorrect assignment of points to the proper cluster. Indeed, this information is crucial for biological data analysis. Thus, for each point, we can check whether the dominant class of the k closer elements allows to correctly predict the actual class of membership (there is no relationship between the value of k used here and that of k -means). Thus, the total number of points correctly classified measures the effectiveness of the clustering at hand. Formally, the error $e_k(D)$ of a k -NN classifier exploiting a the distance matrix among every pair of points. D can be defined as

where N is the total number of points, and $\gamma_k(i)$ is 0 if the predicted class of the i th point (x_i) coincides with its actual class, and 1 otherwise. Low values of the $e_k(D)$ index denote high-quality clusters.

Following [18], we can go deeper in our evaluation by measuring the average number of elements, in a range of k elements (we recall again that we use the expected cluster size value), having the same class as the point under consideration. Practically, we define q_k as the average percentage of points in the k -neighborhood of a generic point belonging to the same class of that point. Formally:

where $Cl(i)$ represents the actual class associated with the i th point in the dataset, $n_i = |Cl(i)|$, and $N_k(i)$ is the set of k points having the lowest distances from x_i, according to the distance used at hand. This value will provide a really interesting information, in fact it will measure the *purity* of the clusters since it take into account the number of points wrongly assigned to a cluster. In principle, a Nearest Neighbor classifier exhibits a good performance when q_k is high. Furthermore, q_k provides a measure of the stability of a Nearest-Neighbor: high values of q_k make a k -NN classifier less sensitive to increasing values k of neighbors considered. The sensitivity of the clustering can also be measured by considering, for a given group of points , the probability that x and y belong to the same class and z belongs to a different class, but z is more similar to x than y is. We denote this probability by $\varepsilon(D)$, which is estimated as

where δ_D is 1 if $D(i,j) < D(i,k)$, and 0 otherwise. This value gives information about the ambiguity in cluster assignments. Here too, low values of $\varepsilon(D)$ denote a good performance of the clustering under consideration.

The results reported in Table 5 and Table 6 show that M-CLUBS produces better results than the other algorithms.

Table 5.

Quality indices for Dataset 1 and Dataset 2.

Method/index	ε	$e_{k=10}$	$q_{k=10}$
Dataset 1			
M-CLUBS	**0.0661**	**0.0984**	**0.9998**
OPTICS	0.1253	0.1976	0.9934
BIRCH	0.1154	0.2010	0.9756
KM++	0.1002	0.1974	0.9803
SMART	0.1086	0.2101	0.9757
DIANA	0.0933	0.1426	0.9846
UPGMA	0.1224	0.1779	0.9811
NN	0.1196	0.1813	0.9803
FN	0.1185	0.1848	0.9794
SiMM-TS	0.0879	0.1065	0.9813
Dataset 2			
M-CLUBS	**0.0054**	**0.0352**	**0.9999**
OPTICS	0.0432	0.1312	0.9875
BIRCH	0.0165	0.0953	0.9923
KM++	0.0487	0.1657	0.9764
SMART	0.0568	0.1789	0.9734
DIANA	0.0113	0.1264	0.9829
UPGMA	0.0197	0.1022	0.9894
NN	0.0201	0.1047	0.9915
FN	0.0188	0.1035	0.9926
SiMM-TS	0.0096	0.067	0.9978

Table options

Table 6.

Quality indices for AD400-10-10 and yeast sporulation.

Method/index	ε	$e_{k=40}$	$q_{k=40}$

AD400-10-10			
M-CLUBS	**0.1041**	**0.0463**	**0.9989**
OPTICS	0.1937	0.0976	0.9713
BIRCH	0.1789	0.1012	0.9668
KM++	0.2046	0.1225	0.9547
SMART	0.2076	0.1317	0.9512
DIANA	0.1216	0.0934	0.9734
UPGMA	0.1814	0.1048	0.9701
NN	0.1515	0.1096	0.9744
FN	0.1546	0.1077	0.9769
SiMM-TS	0.1167	0.0657	0.9932
Yeast sporulation			
M-CLUBS	**0.1534**	**0.2287**	**0.9887**
OPTICS	0.2031	0.2765	0.9755
BIRCH	0.2217	0.2854	0.9689
KM++	0.2431	0.3011	0.9554
SMART	0.2536	0.3046	0.9532
DIANA	0.2176	0.2679	0.9729
UPGMA	0.2189	0.2866	0.9773
NN	0.2245	0.2879	0.9798
FN	0.2263	0.2884	0.9748
SiMM-TS	0.1934	0.2458	0.9843

Table options

Table 5 and Table 6 show that M-CLUBS offers the best performance on all indices and in particular the really high values of q_k (it is practically 1 since it detects exactly the number of clusters for each dataset and the point assignment to cluster is correct) allow to asses that the clusters are well defined, and M-CLUBS outperforms other methods. In measuring e_k and q_k, we used neighborhoods of size closer to the actual cluster size available by datasets provider thus it is a good choice for testing the quality of clusters. The overall structure of the clusters and the points distribution for all datasets (results in Table 5 and Table 6) produced superior performance for M-CLUBS on every index, with particularly low values of ε. This result suggests that M-CLUBS exhibits the highest effectiveness compared to the other approaches *even when SSQ is not the exploited metrics*.

4.3. Evaluating the biological relevance of clusters

In this section we report the experimental results regarding a further comparison we performed to assess the validity of our approach from a biological viewpoint. Indeed, clustering gene expression data is a valid support for functional annotation, tissue classification, regulatory motif identification, and other applications, but choosing the right clustering may be rather difficult. To address this issue, several proposal have been presented such as [12] and [20]. In this paper we exploited the quality measure defined in [20] for biological data clustering evaluation[1] since it summarizes several evaluation metric in a single measure. We ran the experiments in standard mode using all Gene Ontology (GO) classes as input setting of the program and report the obtained *CQS* (Clustering Quality Score) [20]. This analysis assures a stronger validation of the clustering results from a biological viewpoint. The results are reported in Table 7 and state the biological relevance of M-CLUBS is quite high.

Table 7.
Clustering quality score for Dataset 1, Dataset 2, AD400-10-10 and yeast sporulation.

Method/index	CQS
Dataset 1	
M-CLUBS	**30.42**
OPTICS	25.76

BIRCH	26.43
KM++	18.92
SMART	17.79
DIANA	27.71
UPGMA	26.54
NN	24.87
FN	25.03
SiMM-TS	29.32
Dataset 2	
M-CLUBS	**33.47**
OPTICS	27.42
BIRCH	28.02
KM++	24.38
SMART	25.67
DIANA	27.78
UPGMA	27.04
NN	28.65
FN	27.58
SiMM-TS	32.15
AD400-10-10	
M-CLUBS	**34.16**
OPTICS	28.81
BIRCH	29.16
KM++	21.17
SMART	22.43
DIANA	28.36
UPGMA	27.79
NN	29.02
FN	29.65
SiMM-TS	33.59
Yeast sporulation	
M-CLUBS	**27.54**
OPTICS	24.66
BIRCH	25.78
KM++	18.69
SMART	17.55
DIANA	25.22
UPGMA	24.36
NN	24.87
FN	24.62
SiMM-TS	26.91

Table options

The high performance of M-CLUBS also from a biological viewpoint can be understood by considering that it can backtrack on previously wrong computation in the splitting phase. More in detail, by the merging step we can properly assign gene expression to their group, thus to the correct function, when it is the target of the analysis, by updating previous wrong assignment. The latter because, we can group together also “siblings” gene expression in our tree auxiliary structure.

Futher discussion on biological relevance of clustering. In order to further asses the biological coherence of M-CLUBS clusters we briefly discuss here *Enrichment Analysis*. Enrichment Analysis is intended to characterize biological attributes in a given gene set. In this respect the GO dataset is a key resource. In particular, GO ontologies are split into cellular component, molecular function, and biological process. Using these ontologies we can better characterize genes, thus improving the annotation process. Many tools exist for assessing significance of enrichment within a group. They typically exploit hypergeometric testing, but can also be based on a Kolmogorov–Smirnov statistic. These tools usually require empirical estimations of p-values and multiple testing corrections. Due to our peculiar approach, according to [3] and [37], we need to compute for each cluster, the GO annotations and the corresponding p-values, that evaluates the probability that a given cluster occurs. [2] Indeed, we determine whether an observed level of annotation for a group of genes is significant within the context of annotation. For the dataset being analyzed we obtained the following p-values: *Dataset 1* – 4%, *Dataset 2* – 4%, *AD400-10-10* – 3%, *Yeast Sporulation* – 3%. As reported above, such satisfactory results are obtained as we group together “siblings” gene expression when clustering data. As a matter of fact, these results further assess the relevance of our clustering from a biological viewpoint.

5. Conclusion

The naturalness of the hierarchical approach for clustering objects is widely recognized, and also supported by psychological studies of children’s cognitive behaviors [38]. M-CLUBS is providing the analytical and algorithmic advances that have turned this intuitive approach into a data mining method of superior, accuracy, robustness and speed. The speed achieved by our approach is largely due to M-CLUBS’ ability of exploiting the analytical properties of its quadratic distance functions to simplify the computation, thus making M-CLUBS well suited for high sized and high dimensional datasets like the biological ones. We evaluated the effectiveness of our approach by using several method independent quality measures that confirmed the high quality of retrieved clusters by a structural point of view. In particular, the experimental assessment clarified that M-CLUBS guarantees good clusterings for the datasets being analyzed that represent a severe benchmark for biological data scenario. Moreover, we provided a biological interpretation of the clustering solutions by a domain expert and quality measures tailored for biological data that confirmed the high quality of the clusters retrieved by M-CLUBS. We conjecture that similar benefits might be at hand for situations where the samples are in data streams or in secondary store. These situations were not studied in this paper, but represent a promising topic for future research.

Acknowledgments

The authors would like to thank both the anonymous reviewers and IS associate editor who assisted our submission, for their invaluable suggestions and insightful comments which helped us improve the paper significantly. We also thank Irit Gat-Viks and Susmita Datta for providing us the code of their projects and many useful details for using it properly.

Appendix A. Detailed discussion on SSQ reduction

Due to its grid-based divisive agglomerative approach CLUBS produces cluster results of superior quality. Furthermore the algorithm can be executed with superior time and accuracy performances using a simple (and fast) formula that allows us to estimate the SSQ reduction produced by a split. The correctness of the approach that guarantee its convergence is discussed next.

In order to select an optimal binary partition, we need to estimate the SSQ reduction obtained by this split.

Suppose to split a cluster C_s at a given position j, represented by $\langle Q,S,N\rangle$ into two clusters $C_{s'}$ and $C_{s''}$ represented respectively by $\langle Q_1,S_1,N_1\rangle$, and $\langle Q_2,S_2,N_2\rangle$.

Thus, the SSQ reduction is the non-negative value:

$$\Delta SSQ(i,j)=SSQ(C_s)-SSQ(C_{s'})-SSQ(C_{s''}) \quad (2)$$

and is non-negative. For dimension i (for the sake of clarity we denote $SSQ(C_s)$ as $SSQ, SSQ(C_{s'})$ as SSQ_1 and $SSQ(C_{s''})$ as SSQ_2)

$$\Delta SSQ_i(j)=SSQ_i(j)-(SSQ_{1i}(j)+SSQ_{2i}(j))$$

applying formula (1) for *SSQ* we obtain:

we recall that

$Q_i = Q_{1i}(j) + Q_{2i}(j)$

thus we obtain:

by multiplying both terms by and recalling that

we obtain:

Now, the overall ΔSSQ can be obtained by simply summing up the above in all dimensions:

(3)

This formula enables a quick computation of the SSQ difference achieved when a cluster is split, and, equivalently, when two clusters are merged into one. It is straightforward to see that the computation is *deterministic* thus guaranteeing the *termination* of the overall algorithm. More in detail, since it is performed at each cluster split (or merge) in the worst case it will be done $2 \cdot dim$ times (where *dim* is the number of points in the dataset), this correspond to the case that a singleton cluster is generated for each data point and then iteratively each singleton is merged to obtain again the initial dataset. As regards the *correctness* of the algorithm, we recall that cluster topology and assignment strongly rely on the chosen measure, in this respect, our approach is *correct* since it minimize the intra-cluster distances and maximize the inter-cluster distances thus guaranteeing that the obtained partition conforms to cluster definition. In next Section, we will discuss the superior performances of our algorithm and compare it against several clustering approaches.

References

[1] J. Ahn, Y. Yoon, S. Park
Noise-robust algorithm for identifying functionally associated biclusters from gene expression data
Inform. Sci., 181 (3) (2011), pp. 435–449
Loading

[2] M. Ankerst, M.M. Breunig, H.P. Kriegel, J. Sander, Optics: ordering points to identify the clustering structure, in: ACM's Special Interest Group on Management Of Data, 1999, pp. 49–60.
Loading

[3] V. Arnau, S. Mars, I. Marín
Iterative cluster analysis of protein interaction data
Bioinformatics, 21 (3) (2005), pp. 364–378
Loading

[4] D. Arthur, S. Vassilvitskii, k-Means++: the advantages of careful seeding, in: ACM-SIAM Symposium on Discrete Algorithms, 2007, pp. 1027–1035.
Loading

[5] W-H Au, K.C.C. Chan, A.K.C. Wong, Y. Wang
Attribute clustering for grouping, selection, and classification of gene expression data
IEEE/ACM Trans. Comput. Biol. Bioinform., 2 (2005), pp. 83–101
Loading

[6] S. Bandyopadhyay, A. Mukhopadhyay, U. Maulik
An improved algorithm for clustering gene expression data
Bioinformatics, 23 (21) (2007), pp. 2859–2865
Loading

[7] Z. Bar-Joseph, E.D. Demaine, D.K. Gifford, N. Srebro, A.M. Hamel, T. Jaakkola
K-ary clustering with optimal leaf ordering for gene expression data
Bioinformatics, 19 (9) (2003), pp. 1070–1078
Loading

[8] S. Ben-David, M. Ackerman, Measures of clustering quality: a working set of axioms for clustering, in: Neural Information Processing Systems, 2008, pp. 121–128.
Loading

[9] A. Ben-Dor, R. Shamir, Z. Yakhini
Clustering gene expression patterns
J. Comput. Biol., 6 (3–4) (1999), pp. 281–297
Loading

[10] Y.M. Cheung
kmidast-Means: a new generalized k-means clustering algorithm
Pattern Recogn. Lett., 24 (15) (2003), pp. 2883–2893
Loading

[11] S. Chu, J. DeRisi, M. Eisen, J. Mulholland, D. Botstein, P.O. Brown, I. Herskowitz
The transcriptional program of sporulation in budding yeast
Science, 282 (5389) (1998), pp. 699–705
Loading

[12] S. Datta, S. Datta
Evaluation of clustering algorithms for gene expression data
BMC Bioinform., 7 (S-4) (2006)
Loading

[13] D. Defays

An efficient algorithm for a complete link method
Comput. J., 20 (1973), pp. 364–366
Loading

[14] D. DembTIT, P. Kastner
Fuzzy c-means method for clustering microarray data
Bioinformatics, 19 (8) (2003), pp. 973–980
Loading

[15] P. D'haeseleer
How does gene expression clustering work?
Nat. Biotechnol., 23 (12) (2005), pp. 1499–1501
Loading

[16] L.S. Einbond, T. Su, H. A Wu, R. Friedman, X. Wang, A. Ramirez, F. Kronenberg, I.B. Weinstein
The growth inhibitory effect of action on human breast cancer cells is associated with activation of stress response pathways
Int. J. Cancer, 121 (9) (2007), pp. 2073–2083
Loading

[17] M. Ester, H.P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: Knowledge Discovery and Data Mining, 1996.
Loading

[18] S. Flesca, G. Manco, E. Masciari, L. Pontieri, A. Pugliese
Fast detection of XML structural similarity
IEEE Trans. Knowl. Data Eng., 17 (2) (2005), pp. 160–175
Loading

[19] L. Galluccio, O. Michel, P. Comon, A.O. Hero
Clustering with a new distance measure based on a dual-rooted tree
Inform. Sci., 251 (2013), pp. 96–113
Loading

[20] I. Gat-Viks, R. Sharan, R. Shamir
Scoring clustering solutions by their biological relevance
Bioinformatics, 19 (18) (2003), pp. 2381–2389
Loading

[21] J. Gollub, G. Sherlock
Clustering microarray data
Methods Enzymol., 411 (2006), pp. 194–213
Loading

[22] K. Graham, A. De Las Morenas, A. Tripathi, C. King, M. Kavanah, J. Mendez, M. Stone, J. Slama, M. Miller, G. Antoine, H. Willers, P. Sebastiani, C.L. Rosenberg
Gene expression in histologically normal epithelium from breast cancer patients and from cancer-free prophylactic mastectomy patients shares a similar profile
Brit. J. Cancer, 102 (8) (2010), pp. 1284–1293
Loading

[23] I. Gronau, S. Moran, Optimal Implementations of UPGMA and Other Common Clustering Algorithms, Technical Report, 2007.
Loading

[24] P.H. Guzzi, M. Cannataro
mu-CS: an extension of the TM4 platform to manage affymetrix binary data
BMC Bioinform., 11 (2010), p. 315
Loading

[25] P.H. Guzzi, M.T. Di Martino, G. Tradigo, P. Veltri, P. Tassone, P. Tagliaferri, M. Cannataro
Automatic summarisation and annotation of microarray data
Soft Comput., 15 (8) (2011), pp. 1505–1512
Loading

[26] J. Han, M. Kamber
Data Mining: Concepts and Techniques
Morgan Kaufmann (2000)
Loading

[27] N. Heard, C. Holmes, D. Stephens
A quantitative study of gene regulation involved in the immune response of anopheline mosquitoes: an application of bayesian hierarchical clustering of curves
J. Am. Stat. Assoc., 101 (473) (2006), p. 18
Loading

[28] K.A. Heller, Z. Ghahramani
Bayesian hierarchical clustering
Int. Conf. Mach. Learn. (2005), pp. 297–304
Loading

[29] A.K. Jain, M.N. Murty, P.J. Flynn
Data clustering: a review
ACM Comput. Surv., 31 (1999)
Loading

[30] R. Jornsten, B. Yu
Simultaneous gene clustering and subset selection for sample classification via MDL
Bioinformatics, 19 (9) (2003), pp. 1100–1109
Loading

[31] L. Kaufman, P.J. Rousseeuw
Finding Groups in Data: An Introduction to Cluster Analysis
Wiley (2005)
Loading

[32] G. Kerr, H.J. Ruskin, M. Crane, P. Doolan
Techniques for clustering gene expression data
Comput. Biol. Med., 38 (3) (2008), pp. 283–293
Loading

[33] A. Koschmieder, K. Zimmermann, S. Trißl, T. Stoltmann, U. Leser
Tools for managing and analyzing microarray data
Briefings Bioinform., 13 (1) (2012), pp. 46–60
Loading

[34] J.Z.C. Lai, T.J. Huang
An agglomerative clustering algorithm using a dynamic k-nearest-neighbor list
Inform. Sci., 181 (9) (2011), pp. 1722–1734
Loading

[35] R. Liu, L. Jiao, X. Zhang, Y. Li
Gene transposon based clone selection algorithm for automatic clustering
Inform. Sci., 204 (2012), pp. 1–22
Loading

[36] J.B. MacQueen, Some methods for classification and analysis of multivariate observations, in: 5th Berkeley Symposium on Mathematical Statistics and Probability, 1967, pp. 281–297.
Loading

[37] C. Pizzuti, S.E. Rombo
A coclustering approach for mining large protein-protein interaction networks
IEEE/ACM Trans. Comput. Biol. Bioinform., 9 (3) (2012), pp. 717–730
Loading

[38] J.M. Plumert
Flexibility in children's use of spatial and categorical organizational strategies
Recall Develop. Psychol., 30 (5) (1994), pp. 738–747
Loading

[39] C. Rasmussen, B. De La Cruz, Z. Ghahramani, D.L. Wild
Modeling and visualizing uncertainty in gene expression clusters using Dirichlet process mixtures
IEEE/ACM Trans. Comput. Biol. Bioinform. (2007)
Loading

[40] R. Savage, K. Heller, Y. Xu, Z. Ghahramani, W. Truman, M. Grant, K. Denby, D. Wild
R/BHC: fast bayesian hierarchical clustering for microarray data
BMC Bioinform., 10 (1) (2009), p. 242
Loading

[41] R. Sibson
Slink: an optimally efficient algorithm for the single-link cluster method
Comput. J., 16 (1973), pp. 30–34
Loading

[42] R.R. Sokal, F.J. Rohlf
Biometry, the Principles and Practice of Statistics in Biological Research
Freeman, W.H. & Company (1981)
Loading

[43] C.J. Veenman, M.J.T. Reinders
The nearest subclass classifier: a compromise between the nearest mean and nearest neighbor classifier
IEEE Trans. Pattern Anal. Mach. Intell., 27 (9) (2005), pp. 1417–1429
Loading

[44] W. Wang, J. Yang, R.R. Muntz, Sting: a statistical information grid approach to spatial data mining, in: Very Large Data Bases, 1997, pp. 186–195.
Loading

[45] W. Wang, J. Yang, R.R. Muntz
An approach to active spatial data mining based on statistical information
IEEE Trans. Knowl. Data Eng., 12 (5) (2000), pp. 715–728
Loading

[46] K.Y. Yeung, D.R. Haynor, W.L. Ruzzo
Validating clustering for gene expression data
Bioinformatics, 17 (4) (2001), pp. 309–318
Loading

[47] T. Zhang, R. Ramakrishnan, M. Livny
Birch: a new data clustering algorithm and its applications
Data Min. Knowl. Discov., 1 (2) (1997), pp. 141–182
Loading

Corresponding author. Tel.: +39 0984831735; fax: +39 0984839054.

1 We thank Irit Gat-Viks and Susmita Datta for providing us the code of their projects and many useful details for using it properly.

2 The software is available at http://www.search.cpan.org/dist/GO-TermFinder/.