OPEN ACCESS Freely available online

PLOS ONE

SMART: Unique Splitting-While-Merging Framework for Gene Clustering

Rui Fa[1]*, David J. Roberts[2,3], Asoke K. Nandi[1,4]*

1 Department of Electronic and Computer Engineering, Brunel University, Uxbridge, Middlesex, United Kingdom, **2** National Health Service Blood and Transplant, Oxford, United Kingdom, **3** The University of Oxford, John Radcliffe Hospital, Oxford, United Kingdom, **4** Department of Mathematical Information Technology, University of Jyväskylä, Jyväskylä, Finland

Abstract

Successful clustering algorithms are highly dependent on parameter settings. The clustering performance degrades significantly unless parameters are properly set, and yet, it is difficult to set these parameters *a priori*. To address this issue, in this paper, we propose a unique splitting-while-merging clustering framework, named "splitting merging awareness tactics" (SMART), which does not require any *a priori* knowledge of either the number of clusters or even the possible range of this number. Unlike existing self-splitting algorithms, which over-cluster the dataset to a large number of clusters and then merge some similar clusters, our framework has the ability to split and merge clusters automatically during the process and produces the the most reliable clustering results, by intrinsically integrating many clustering techniques and tasks. The SMART framework is implemented with two distinct clustering paradigms in two algorithms: competitive learning and finite mixture model. Nevertheless, within the proposed SMART framework, many other algorithms can be derived for different clustering paradigms. The minimum message length algorithm is integrated into the framework as the clustering selection criterion. The usefulness of the SMART framework and its algorithms is tested in demonstration datasets and simulated gene expression datasets. Moreover, two real microarray gene expression datasets are studied using this approach. Based on the performance of many metrics, all numerical results show that SMART is superior to compared existing self-splitting algorithms and traditional algorithms. Three main properties of the proposed SMART framework are summarized as: (1) needing no parameters dependent on the respective dataset or *a priori* knowledge about the datasets, (2) extendible to many different applications, (3) offering superior performance compared with counterpart algorithms.

Citation: Fa R, Roberts DJ, Nandi AK (2014) SMART: Unique Splitting-While-Merging Framework for Gene Clustering. PLoS ONE 9(4): e94141. doi:10.1371/journal.pone.0094141

Editor: Sergio Gómez, Universitat Rovira i Virgili, Spain

Received November 6, 2013; **Accepted** March 14, 2014; **Published** April 8, 2014

Funding: The authors are thankful for the financial support from National Institute for Health Research (NIHR), UK. The project reference is NIHR-RP-PG-0310-1004-AN. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing Interests: The authors have declared that no competing interests exist.

* E-mail: Rui.Fa@brunel.ac.uk (RF); asoke.nandi@brunel.ac.uk (AKN)

Introduction

Clustering methods have been widely used in many fields, including biology, physics, computer science, communications, artificial intelligence, image processing, and medical research, requiring analysis of large quantities of data to explore the relationships between individual objects within the respective datasets [1–13]. However, clustering is one of the most difficult and challenging problems in the realm of machine learning due to the lack of universal and rigorous mathematical definition. The definition of clustering often depends on the specific systems or problems, e.g. in computer vision, where it is defined as image segmentation [1,2], or in complex network analysis, where it is known as graph clustering or community detection [14–16].

After some pioneering works by Eisen et al. [6], Golub et al. [7], and Tamayo et al. [8], clustering was extensively employed in gene expression analysis where microarray and real time sequencing have allowed rapid measurement of genome-wide transcription[3,5,17–25]. There are many families of clustering algorithms used in the gene expression analysis, including partitional clustering, hierarchical clustering, model-based clustering, self-organizing clustering [3,23]. Results of most of successful clustering algorithms strongly depend on the determined number of clusters, e.g. k-means, model-based clustering, and hierarchical clustering (when the clustering memberships need to be determined). However, in many cases, *a priori* knowledge of the actual number of clusters is not available. Thus, the number of clusters has to be estimated beforehand. The problem of determining the best number of clusters needs to be addressed in another branch of research in clustering analysis, known as clustering validation [26–28]. Among various clustering validation criteria, clustering validity indices, also known as relative criteria, have been employed to quantitatively evaluate the goodness of a clustering result and estimate the best number of clusters. There are two main classes of validity indices: a) model-based or information theoretic validation, e.g. minimum description length (MDL) [29], minimum message length (MML) [30,31], Bayesian information criterion (BIC) [32], Akaike's information criterion (AIC) [33], and the normalized entropy criterion (NEC) [34]; b) geometric-based validation, which considers the ratio of within-group distance to between-group distance (or its reciprocal), such as Calinski-Harabasz (CH) index [35], *Dunn's* index (DI) [36], Davies-Bouldin (DB) index [37], I index [38], Silhouette index (SI) [39], the geometrical index (GI) [40], the validity index V_I [41] and the parametric validity index (PVI) [42,43].

Once an appropriate clustering validity index is selected, the general practice for determining the best number of clusters has few steps: a set of clustering results are firstly obtained by a clustering algorithm with fixed number of clusters within a predetermined range $[K_{min},K_{max}]$; then, these clustering results are evaluated by the chosen validity index; finally, depending on the chosen validity index, maximum or minimum index value indicates the best number of clusters (in some cases if the index value has an increase or decrease trend against the number of cluster, the significant knee point indicates the best number of clusters). However, this solution requires an extensive search for the number of clusters and is tedious work for large number of clusters

Moreover, the initialization of clustering is also a major issue. For some algorithms with the deterministic initialization, e.g. hierarchical clustering and clustering with kauffman approach initialization (KA) [44], the optimal solution is not always guaranteed. For some algorithms sensitive to initialization, such as k-means with random initialization, expectation-maximization (EM) [17], and self-organization map (SOM) [45], they may get stuck at local minimum. Addressing this problem requires running the algorithm repeatedly with the same dataset using several different initializations. This makes such clustering algorithms more computationally unfavourable. Thus, better options would be integrative frameworks or strategies which provide an automatic and consistent clustering, so users do not have to worry about setting those data-specific parameters.

Earliest attempts of automated clustering without employing any *a priori* knowledge of number of clusters were growing cell structure [46] and growing neural gas [47]. Although these algorithms are useful to visualize high dimensional data, they are not suitable for clustering because they over-fit the data. A self-splitting competitive learning (SSCL) algorithm was proposed to achieve the automated clustering [48]. In SSCL, a competitive learning paradigm, so called *one-prototype-take-one-cluster* (OPTOC), was developed for self-splitting by employing an asymptotic property vector (APV) to guide the learning of a prototype; meanwhile a split validity criterion was embedded in SSCL to assess whether each cluster would contain more than one prototype: if it was the case, then cluster would be split into two. However, there are two vital issues to prevent its practical uses: 1) the prototypes are easily trapped into global centroid, especially the first few ones [48], and 2) the parameters for stopping both OPTOC learning and splitting are crucial to the algorithm but they are difficult to estimate reliably [49]. Yet, the SSCL has an attractive advantage in that it does not require *a priori* knowledge about the number of clusters in the input dataset.

Another strategy for automated clustering has been proposed using a similar method [49–52]. In these approaches, the input data was over-clustered to a large number of partitions, say k_{max}, then these partitions were merged to fewer clusters, which were closer to the natural clusters. This strategy is called splitting-then-merging (STM). In terms of clustering techniques, the algorithm by Figueiredo and Jain [50] was based on unsupervised learning of finite mixture models (ULFMM), the self-splitting-merging competitive learning (SSMCL) by Wu and colleagues in [49] was based on OPTOC competitive learning paradigm, and a variational Bayesian Gaussian mixtures (VBGM) framework has been explored [51,52]. Another critical difference between these algorithms is that the criteria for selecting final clustering are different. In ULFMM, along with the merging process from k_{max} to k_{min}, a model order selection criterion, which was minimum message length (MML) in their case, was used; in SSMCL, as a merging criterion was defined according to the measurement of distortion between two clusters, merging process would not stop until no cluster met the merging criterion; in VBGM, after the convergence of the optimization algorithm, the estimated number of clusters tends to be the number of non-empty clusters. There are two critical issues in the STM framework: one is that the maximum number of clusters k_{max} has to be determined *a priori*, however such an upper limit is subjective and sometimes only an inexact estimate is available; another issue is that as one of bottom-up algorithms, the STM framework cannot produce a very accurate clustering result in some circumstances, since it makes clustering decisions based on local patterns without initially taking into account the global distribution. Recently, Mavridis and colleagues proposed a parameter-free clustering (PFClust) algorithm, which is able to determine the number of clusters automatically [53]. PFClust clusters the dataset in two steps: first step is to estimate expectation and variance of intra-cluster similarity by randomisation; second step is to cluster the dataset based on the threshold calculated in randomisation. However, to select a suitable threshold, PFClust needs a good approximation to the distribution of mean intra-cluster similarities, and it requires a large number of randomisation which is time-consuming.

Here, we propose a new splitting-merging clustering framework, named "splitting-merging awareness tactics" (SMART) to overcome these problems. The proposed framework is different from aforementioned over-cluster-then-merge strategy and employs a novel splitting-while-merging (SWM) strategy. The proposed system integrates such crucial clustering techniques as cluster splitting methods, cluster similarity measurement, and clustering selection, within a framework to mimic human perception doing the sorting and grouping, which was inspired by the work of Zhang and Liu [48]. The framework starts with one cluster and accomplishes many clustering tasks to split and merge clusters. While splitting, a merging process is also taking place to merge the clusters which meet the merging criterion. In this process, SMART has the ability to split and merge clusters automatically in iterations. Once the stop criterion is met, the splitting process terminates and then a clustering selection method is employed to choose the best clustering from several generated ones. Moreover, the SMART framework is not restricted to a specific clustering technique. In this paper, we implement SMART in two algorithms using two distinct clustering paradigms: SMART I employs OTPOC competitive learning as the splitting algorithm and the calculation of cohesion between two clusters [54] as the merging criterion; and SMART II employs modified component-wise expectation maximization of mixtures (CEM^2) [50], which was originally proposed in [55], to fulfil splitting and merging. For both algorithms, once the splitting-merging process terminates, a model order selection algorithm plays a critical role in selecting the best clustering among the generated clusterings during the splitting procedure. Two benchmark demonstration datasets are used to illustrate each step in the SMART flow. The main purpose of this paper is to develop the SMART framework and its algorithms for microarray gene expression datasets. Thus, two simulated gene expression datasets and two real microarray gene expression datasets are studied using SMART. By comparing the performance of several metrics, namely adjusted Rand index (ARI) [61,62], correct selection rate (CSR) of number of clusters, the estimated number of clusters ($\hat{K}$), normalized mutual information (NMI), Jaccard index (JI), Silhouette index (SI), Calinski-Harabasz (CH) index, and minimum message length (MML), the numerical results show that our proposed method is superior. Most importantly, SMART does not require any parameters dependent on the respective dataset or *a priori* knowledge about the datasets.

The main sections of this paper are organised in the following sequence. The next section describes the philosophy of the proposed framework. We then provide the results of many examples, including two demonstration examples, two simulated datasets and two real gene expression datasets, to support the proposed framework. Subsequently, the clustering techniques employed in the SMART framework are detailed in Methods section. Finally, we conclude with a discussion of applications for future research.

Results

SMART Framework

First of all, we must emphasize that SMART is a framework rather than a simple clustering algorithm, within which a number of clustering techniques are organically integrated. Thus, conceptually, SMART does not fall into any categories classified in [2,4]. In this section, we focus on the overview of the whole framework, and describe implementation solutions and specific clustering techniques in the following sections.

Suppose that we are going to partition the dataset $\mathcal{X}=\{\boldsymbol{x}_i|1\leq i\leq N\}$, where $\boldsymbol{x}_i\in\mathbb{R}^{M\times 1}$ denotes the i-th object, M is the dimension, and N is the number of objects. The flowchart of the framework is illustrated in Fig. 1.

The whole clustering procedure is divided into four tasks. SMART starts with one cluster ($K=1$, where K is the number of clusters), and the cluster needs to be initialized, which is Task 1. Subsequently, the data goes through a SWM process, where splitting and merging are automatically conducted in iterations. In the splitting step of each iteration, which is labelled Task 2, SMART splits one of the clusters into two. After a splitting step, the new clustering is censored by a merging criterion, which is associated with Task 3. If the condition for merging is satisfied, then one merges the two clusters, otherwise the merging step is skipped. SMART then goes through a termination-check, where a stopping criterion is applied. If the condition for termination is not satisfied, SMART goes to the next iteration and continues to split, otherwise, SMART finishes the splitting-merging process. The last step is the clustering selection (Task 4).

Note that these tasks in the SMART flow can be completed using many clustering techniques in the literature, e.g., Task 1 can be done by any initialization technique either deterministic or random; Tasks 2 and 3 may be achieved by any splitting algorithm and merging criterion respectively or they may be combined into one algorithm; and Task 4 can be accomplished by any of either model order selection algorithms or validity indices. Different techniques will make the implementation slightly different but the flow does not change. Moreover, different clustering algorithms bring different features into the framework and so SMART can be customized for different applications. In the following Methods section, we will develop two SMART algorithms using different splitting and merging algorithms, i.e., OPTOC competitive learning and finite mixture model learning, which are called SMART I and SMART II, respectively, and they have similar configurations. In particular, both use MML [30,31] as clustering selection algorithm and use the same termination criterion in the SWM process, namely the maximum number of merges, N_{max}. The logic behind the termination criterion is that normally merging will not start until optimal clustering is reached. Once N_{max} is reached, the splitting and merging will terminate automatically. We summarise the categorization of existing self-splitting-merging algorithms and our two SMART algorithms in Table 1. All existing self-splitting-merging algorithms employ the STM strategy with different clustering paradigms; instead our

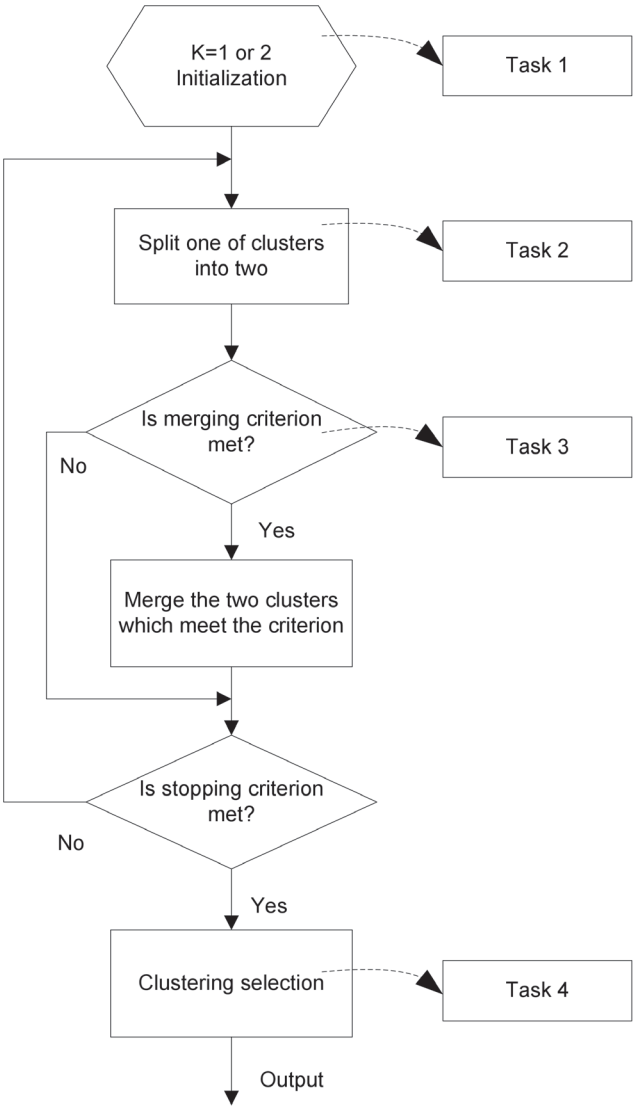

Figure 1. The flow chart of the SMART framework. SMART is initialized in Task 1; SMART splits one of clusters into two in Task 2; the new clustering is censored by a merging criterion in Task 3; SMART goes through the SWM process iteratively and generates many candidate clusterings; finally, the optimal clustering is selected by clustering selection criterion in Task 4.
doi:10.1371/journal.pone.0094141.g001

SMART algorithms employ the SWM strategy. For the purposes of direct comparisons with the existing STM algorithms, we propose two specific SMART algorithms. Nevertheless, it should

Table 1. Categorisation of two existing splitting-then-merging (STM) algorithms and our two splitting-while-merging (SWM) SMART algorithms.

	STM (requiring K_{max})	SWM
Competitive Learning	SSMCL	SMART I
Finite Model Mixtures (Gaussian)	ULFMM, VBGM	SMART II

doi:10.1371/journal.pone.0094141.t001

Table 2. The list of Software with which all clustering methods in this paper are implemented.

Methods	Software	Reference
MFA	MATLAB	[22]
MCFA	MATLAB	[21]
SSMCL	MATLAB	[49]
ULFMM	MATLAB (Downloaded)	[50]
VBGM	MATLAB (Downloaded)	[52]
SMART I	MATLAB	-
SMART II	MATLAB	-
DBSCAN	R (FPC Package)	[60]
MCLUST	R (Mclust Package)	[17]
PFClust	Java (downloaded)	[53]

doi:10.1371/journal.pone.0094141.t002

be noted that, within the proposed SMART framework, many other algorithms can be derived for different clustering paradigms.

Experiment Set-up

In this paper, we use two demonstration datasets, which are bivariate mixture models: the first one is quadrature phase-shift keying (QPSK) data with signal-to-noise ratio (SNR) equal to 15 dB and the second one is a 3-component bivariate mixture [50]. Since we have more interest in the microarray gene expression data analysis, we employ two microarray gene expression data modelling methods to simulate or synthesize gene expression data. One simulates the state-based gene expression data [19] and another one simulates the periodic behaviour of yeast cell cycle[17,59]. The advantages of using simulated data are that the ground truth is known and we have the freedom to manipulate the noise level of the data by tuning a few parameters. Additionally, two real microarray gene expression datasets are studied using SMART. The performance comparisons are carried out between the SMART algorithms and both SSMCL, ULFMM, VBGM, DBSCAN [60], MCLUST [17] and PFClust [53] in all experiments. Moreover, two state-of-the-art mixture model clustering, namely the mixture of factor analysers (MFA) [22] and the mixture of common factor analysers (MCFA) [21] are compared. Since these algorithms require a time-consuming exhaustive search over both a range of number of clusters (K) and a range of number of factors (q), with a number of initial starts, we only compare them in real datasets. We list the software in which all clustering algorithms were implemented in Table 2. In our study, many metrics are investigated: ARI, CSR of number of clusters, the estimated number of clusters $\hat{K}$, NMI, JI, SI, CH and MML, where both the mean and the standard deviation are presented for ARI, $\hat{K}$, NMI, JI, SI, CH and MML. Note that for all metrics except $\hat{K}$ and MML, the maximal values are the measures of the best clustering results. CSR is the ratio of the times of the number of clusters being correctly selected, to the total number of experiments. In the following experiments, the parameters for SMART I and II are set as: $N_{max}=5$ for both SMART I and II; $k_{max}=30$ for SSMCL, ULFMM and VBGM. For MFA and MCFA, the parameters are set as: $k_{min}=2$, $k_{max}=30$, the number of factors q from 1 to 10, using 50 initial starts. For PFClust, we set the number of randomisation to be 10000. For MCLUST, we employ MML as clustering validation to estimate the number of clusters because it does not estimate the number of clusters automatically. For all demonstration datasets and simulated datasets, we feed them into clustering algorithms as they were generated without normalisation. Thus, the inputs for all

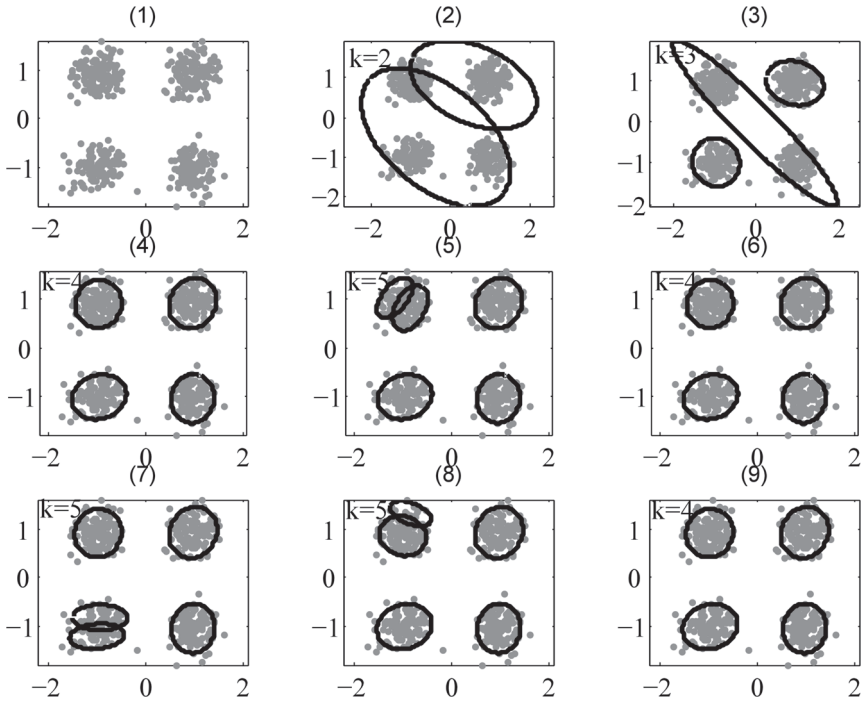

Figure 2. The demonstration of SMART I using QPSK dataset in D1 example. Sub-figures(1) – (8) demonstrate that the procedure of SMART I (SWM process). It starts with $K=1$ (sub-figure(1)), splits into $K=2$. $K=3$, $K=4$ and $K=5$ shown sub-figures(2) – (5) respectively, and then merges some clusters while splitting as shown in sub-figures(6) – (8). The sub-figure(9) is the final clustering result. Parameter settings: $T_{chs}=20$ and $N_m=5$.
doi:10.1371/journal.pone.0094141.g002

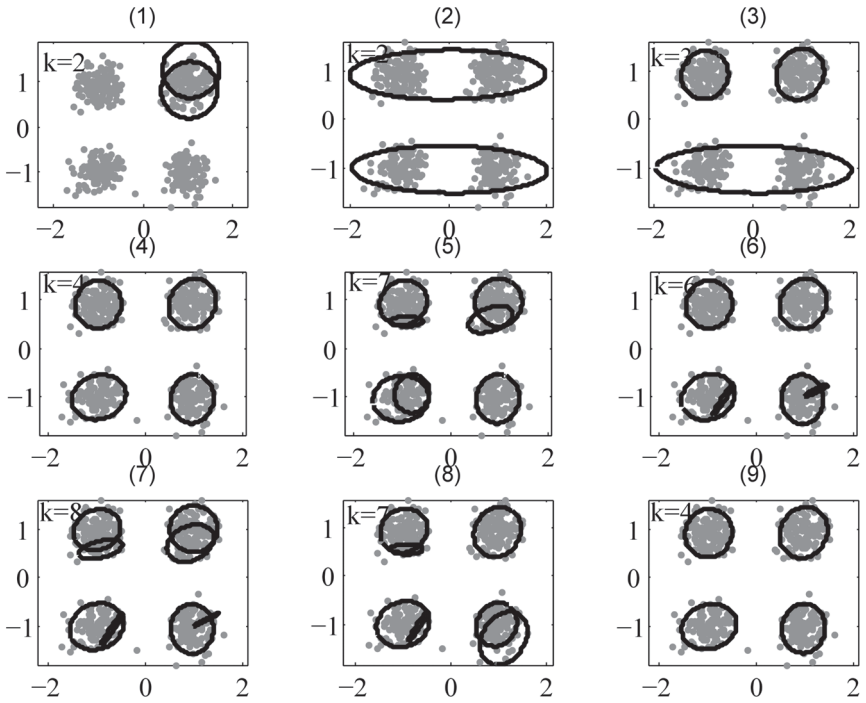

Figure 3. The demonstration of SMART II using QPSK dataset in D1 example. Sub-figures (1) – (8) demonstrate the procedure of SMART II. It starts with $K=2$ (sub-figure(1)), splits into $K=2$. $K=3$, $K=4$ and $K=7$ shown sub-figures(2) – (5) respectively, and then merges some clusters while splitting as shown in sub-figures(6) – (8). Sub-figure(9) is the final clustering result. Parameter setting: $N_m=5$.
doi:10.1371/journal.pone.0094141.g003

algorithms are treated equally. Although we standardise each profile of gene to be zero mean and unit variance for real datasets, it is still the case that the inputs for all algorithms are treated equally.

Demonstration Examples

In the first place, we employ a benchmark test dataset – 512-samples QPSK data with SNR level of 15 dB, which is labelled D1 dataset. This dataset can also be viewed as a 4-component Gaussian mixture. This example may clearly demonstrate how SMART I and II work, as shown in Figs. 2 and 3, respectively. In both Figs. 2 and 3, subfigures from (1) to (8) illustrate the proposed SWM process in the SMART framework, and subfigure (9) shows the final clustering result. The results show that the first merge of SMART I is after $K=5$ shown in Fig. 2-(5) and the first merge of SMART II is after $K=5$ shown in Fig. 3-(5). Subsequently, the merge counter measures the times of merges until the SWM process terminates. To compare SMART with the state-of-the-art clustering algorithms, namely SSMCL, ULFMM, DBSCAN, MCLUST, PFClust and VBGM, using the same dataset, we repeat the clustering experiments 1000 times for each algorithm. The numerical results for D1 are shown in Table 3. SMART II, DBSCAN, MCLUST, PFClust and VBGM produce perfect results in all metrics, which means that there is no mis-clustered members at all in their results in the whole experiment. For other algorithms, the metrics are not always consistent. SSMCL has the poorest performance compared with other algorithms according to all metrics except that it has lower MML value than SMART I. SMART I provides higher values of CSR, SI, and CH, and has more closer mean and smaller standard deviation of $\hat{K}$ than ULFMM, but ULFMM has better performance in ARI, NMI, JI, and MML. The reason for this observation may be that SSMCL occasionally put some objects into wrong clusters but the number of clusters is correct, while ULFMM sometime wrongly splits an actual cluster into two but the objects are mostly in the correct clusters.

The second demonstration example D2 is a 3-component bivariate Gaussian mixture dataset used in [50], whose mixture probabilities are $\alpha_1=\alpha_2=\alpha_3=1/3$, with mean vectors at $[0,-2]^T$,$[0,0]^T$,$[0,2]^T$, and equal covariance matrices $\mathrm{diag}\{4,0.4\}$. The covariance matrices are $\mathrm{diag}\{2,0.2\}$ in [50], but we double them in our study as we try to discern the best algorithm by enlarging the differences among their performances. The numerical results for D2 dataset are shown in Table 4. SSMCL and SMART I fail in this experiment. The reason is that the competitive learning is a spherical or hyper-spherical algorithm so it is not suitable for the clustering of elliptical or hyper-elliptical datasets. Although SMART I has higher CH and SI values than both SMART II and ULFMM, other metrics all reveal that SMART I performs poorly. SMART II has 100% CSR in the experiment and other performance in all metrics are best except CH and SI. The explanation of this observation is that CH and SI use Euclidean distance, which is a hyper-spherical metric. Thus CH and SI are not reliable in this case. It is also worth noting that VBGM has much poorer performance than SMART II in this case, in particular, only 72.4% CSR. These results reflect that SMART II is much better than ULFMM and VBGM where there is considerable noise. DBSCAN fails in this experiment can does not cluster at all (resulting all-zero partition); MCLUST and PFClust perform poorly in this dataset. The clustering procedures of SMART II and ULFMM are shown in Fig. 4 and 5, respectively. These two demonstration examples show how the mechanism of SMART is working. To some extent, they also show that the SMART framework is more effective and more practical than ULFMM, because it is not necessary for SMART to set k_{max}.

Table 3. Performance comparison of many metrics, including ARI, CSR, $\hat{K}$, NMI, JI, SI, CH, MML, for all algorithms in D1.

Algorithms	ARI	CSR	$\hat{K}$	NMI	JI	SI	CH	MML
SSMCL	0.993±0.03	70.2%	4.3±0.5	0.995±0.03	0.993±0.05	0.902±0.04	1.2E3±147	779.9±24.4
ULFMM	0.998±0.01	93.2%	4.1±0.5	0.998±0.01	0.998±0.01	0.898±0.10	1.26E3±112	779.8±2.0
VBGM	**1±0**	**100%**	**4.0±0**	**1±0**	**1±0**	**0.923±0**	**1.29E3±0**	**779.6±0**
DBSCAN	**1±0**	**100%**	**4.0±0**	**1±0**	**1±0**	**0.923±0**	**1.29E3±0**	**779.6±0**
MCLUST	**1±0**	**100%**	**4.0±0**	**1±0**	**1±0**	**0.923±0**	**1.29E3±0**	**779.6±0**
PFClust	**1±0**	**100%**	**4.0±0**	**1±0**	**1±0**	**0.923±0**	**1.29E3±0**	**779.6±0**
SMART I	0.994±0.02	98.6%	4.0±0.2	0.996±0.03	0.995±0.02	0.918±0.05	1.28E3±125	782.2±38.3
SMART II	**1±0**	**100%**	**4.0±0**	**1±0**	**1±0**	**0.923±0**	**1.29E3±0**	**779.6±0**

doi:10.1371/journal.pone.0094141.t003

Simulated Gene Expression Datasets

The first experiment (S1) is a stochastic model which simulates the state-based gene expression data [19]. There are 11 clusters $\{\mathcal{C}_k|k=1,...,11\}$ of genes with $M=50$ samples in the simulated data. The cluster size $n_k(k=1,...,11)$ satisfy Poisson distribution $n_k \sim 4\times \mathrm{Pois}(\lambda)$. The expression values are simulated as a hierarchical log-normal model in each cluster. For $\mathcal{C}_k$, firstly, a vector of cluster template for the cluster is created with four periods of expression of size $m_p(p=1,...,4)$. The sizes of m_p are from a uniform distribution such that $\sum m_p = M$ and $m_p > 2$. The initial template in four periods is simulated from $\log(\mu_p^{(k)}) \sim N(\mu,\sigma^2)$. Secondly, sample variability (σ_s^2) is introduced and the gene sample template $G_j^{(k)}(j=1,...,11)$ is generated from $\log(G_j^{(k)}) \sim N(\mu_p^{(k)},\sigma_s^2)$, where j is such that $(m_1+...+m_{p-1}) < j \le (m_1+...m_p)$. Then for each gene vector i in sample j, the gene variability is added and expression values are generated as $\log(x_{ij}) \sim N(\log(G_j^{(k)}),\sigma_0^2)$. Lastly, once gene data is simulated, a random noise from normal distribution $(\sigma_n = 0.05,0.1,0.2,0.4,0.8,$ and $1.2)$ is added. The parameters used in this model are set as: $\mu=6,\sigma=1,\sigma_s=1.0,\sigma_0=0.1$, and $\lambda=10$. We generate 100 datasets for each σ_n.

The errorbar charts of ARI, JI, CSR, and NMI are shown in Figs. 6 (a) – (d) respectively. Generally speaking, in S1, it is found that the FMM clustering works better than the competitive learning and that the SMART framework has better performance than over-cluster-then-merge strategy. The proposed SMART II algorithm has superior performance when the noise level is low or moderate. It has above 60% CSR and ARI, JI, and NMI values close to 1 when the noise variance σ_n is equal or smaller than 0.1. In all noise levels where σ_n is below 0.4, SMART II always provides the superior performance among the compared algorithms and no algorithm works well when σ_n is greater than 0.4. We also investigate the impact of the parameter N_{max} on the performance of SMART in S1 datasets, and the results are shown in Fig. 7. It is worth noting that the performance of SMART is stable when N_{max} is greater than or equal to two; in other words, the performance of SMART is not sensitive to the value of N_{max}.

In the second simulated dataset experiment (S2), we employ the method in [59] to generate a number of synthetic gene expression datasets with 500 synthetic genes in each dataset and 24 samples for each gene. These 500 genes belong to $K=5$ clusters and each cluster has 100 members. The model of cyclic gene expression is given by

$$x_{ij} = r + [l+pr](r + [l+pr]\sin(2\pi j/8 - \omega_i + qr), \qquad (1)$$

where x_{ij} is the expression value of the i-th gene at the j-th time point, each instant of r is an independent random number from the standard normal distribution $\mathcal{N}(0,1)$, the parameter l controls the magnitude of the sinusoid and it is fixed to three here. The parameter p controls the random component added to the magnitude and the parameter q controls the random component added to the phase. The parameter ω_i is the phase shift of the i-th gene and will determine which cluster the gene i will be in. Since the noise in this model is not additive, we have to couple p and q to be a pair, and raise both their values to change the noise power. By increasing values of p and q will increase the noise power increases. The paired parameters are listed as $(p,q)\in\{(0.1,0.01)$, $(0.3,0.03)$, $(0.5,0.05)$, $(0.7,0.07)$, $(0.9,0.09)$, $(1.1,0.11)$, $(1.3,0.13)$, $(1.5,0.15)$, $(1.7,0.17)$, $(1.9,0.19)$, $(2.1,0.21)$, $(2.3,0.23)$, $(2.5,0.25)\}$. Thus, there are 13 parameter pairs (PPs) from PP1 to PP13 representing 13 noise levels from low to high. For each

Table 4. Performance comparison of many metrics, including ARI, CSR, $\hat{K}$, NMI, JI, SI, CH, MML, for all algorithms in D2.

Algorithms	ARI	CSR	$\hat{K}$	NMI	JI	SI	CH	MML
SSMCL	0 ± 0	0.0%	1 ± 0	0 ± 0	0 ± 0	/	/	/
ULFMM	0.64 ± 0.03	88.1%	3.2 ± 0.7	0.61 ± 0.02	0.61 ± 0.02	0.27 ± 0.11	138.3 ± 21.4	3.68E3 ± 5.6
VBGM	0.60 ± 0.05	72.4%	3.77 ± 1.3	0.49 ± 0.04	0.44 ± 0.05	0.23 ± 0.05	134.3 ± 16.6	3.70E3 ± 17.4
DBSCAN	-	-	-	-	-	-	-	-
MCLUST	0.25 ± 0.0	0.0%	12 ± 0.0	0.44 ± 0.0	0.23 ± 0.0	0.40 ± 0.0	304.2 ± 0.0	3.78E3 ± 0.0
PFClus	0.16 ± 0.0	0.0%	2.0 ± 0.0	0.15 ± 0.0	0.33 ± 0.0	0.33 ± 0.0	197.3 ± 0.0	3.80E3 ± 0.0
SMART I	0.19 ± 0.15	14.6%	3.3 ± 1.4	0.23 ± 0.17	0.33 ± 0.06	**0.43 ± 0.06**	**412.4 ± 70.3**	3.72E3 ± 263.5
SMART II	**0.70 ± 0.01**	**100%**	**3.0 ± 0**	**0.62 ± 0.01**	**0.63 ± 0.01**	0.30 ± 0.003	145.8 ± 0.2	**3.68E3 ± 0.2**

doi:10.1371/journal.pone.0094141.t004

pair of parameters, we generate 100 datasets, and subsequently, we get 100 clustering results from each clustering algorithm. Figs. 8 (a) – (d) respectively show the errorbar charts of ARI, JI, CSR, and NMI achieved by each method in the S2 experiment. The results lead to the similar conclusion obtained in S1 experiment, which is namely FMM clustering works better than competitive learning and the SMART framework has better performance than over-cluster-then-merge strategy. The most impressive observation is that the proposed SMART II algorithm shows all ARI, JI and NMI values equal to one and 100% CSR until the 7-th PP, which is (1.3,0.13), while no other method has 100% CSR performance and no other method has comparable performance in the whole experiment. We carry out the same investigation of impact of the parameter N_{max} as in the S1 datasets. The results are shown in Fig. 9, which also indicate that the performance of SMART is not sensitive to the value of N_{max}.

Real Microarray Gene Expression Datasets

Although the simulated experiments may have the advantage that they show different performance in different conditions for each method, they suffer the crucial drawback that they are not real. So we have tested our SMART using real datasets.

The first real dataset (R1) is a subset of the leukemia dataset [7], which consists of 38 bone marrow samples obtained from acute leukemia patients at time of diagnosis. There are 999 genes in the dataset [63]. The biological truth is that the samples include 3 groups: 11 acute myeloid leukemia (AML) samples, 8 T-lineage acute lymphoblastic leukemia (ALL) samples and 19 B-lineage ALL samples [7,63,64]. We repeat the clustering experiments 1000 times for each method. We also compare two state-of-the-art mixture model clustering algorithms, namely MFA and MCFA, with our proposed SMART algorithms. Since these algorithms require a time-consuming exhaustive search over a range of K and a range of q with a number of initial starts, we run them only once for each K and each q with 50 initial starts, where K ranges from 2 to 30 and q ranges from 1 to 10. The results are shown in Table 5. SSMCL and VBGM totally fail in this experiment, where SSMCL always converges to one cluster and VBGM always terminates at $K_{max} = 30$. Impressively, SMART I has significantly better performance than ULFMM and has nearly 30% greater CSR and better performance in other metrics. In terms of mean and standard deviation of $\hat{K}$, SMART I has a mean closer to the true value and significantly smaller standard deviation than ULFMM. Both MFA and MCFA have their lowest MML values with three clusters, but compared with two SMART algorithms, they show poorer performance in all metrics. SMART II has the superior performance and always provides 100% CSR and best performance in all other metrics. Particularly, SMART II also has very small variations in these metrics, that is, it provides consistent results even though it is randomly initialized. In this experiment, DBSCAN, MCLUST, and PFClust perform poorly and do not provide the correct estimates of the true number of clusters. Furthermore, their other validation metrics are worse than the SMART II algorithm. We have also examined the impact of variable values of N_{max} on the performance. We choose three values for the testing, $N_{max} = 5$, 10, and 20. The results are shown in Table 6. We can read from the Table that in all performance metrics, there is no significant difference among the results from different N_{max} values. Thus, It confirms again that the SMART algorithms are not sensitive to the parameter N_{max} in this test.

Another real dataset (R2) is yeast cell cycle α-38 dataset provided in Pramila *et al.* [65]. It consists of 500 genes with highest periodicity scores and each gene has 25 time samples. Additionally, their peaking times as percentages of the cell cycle have also

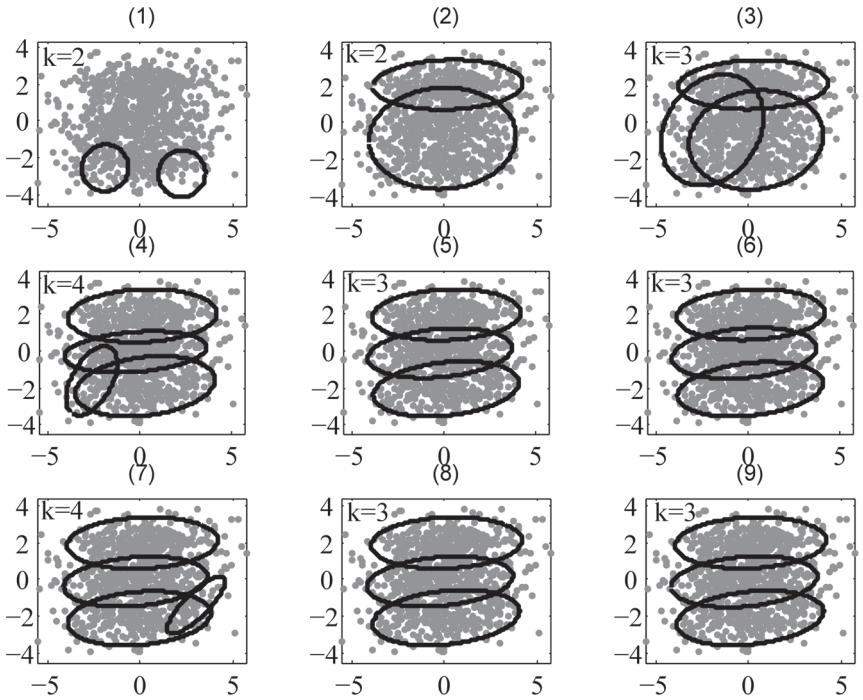

Figure 4. The demonstration of SMART II using Gaussian mixture dataset in D2 example. Sub-figures (1) – (8) demonstrate the procedure of SMART II. SMART II starts from $K=2$ as shown in sub-figures(1) and (2), splits the dataset to $K=3$ and $K=4$ shown in sub-figures(3) and (4); the merging commences while splitting continues as shown in sub-figures(5) – (8). Sub-figure(9) is the final clustering result. Parameter setting: $N_m=5$.
doi:10.1371/journal.pone.0094141.g004

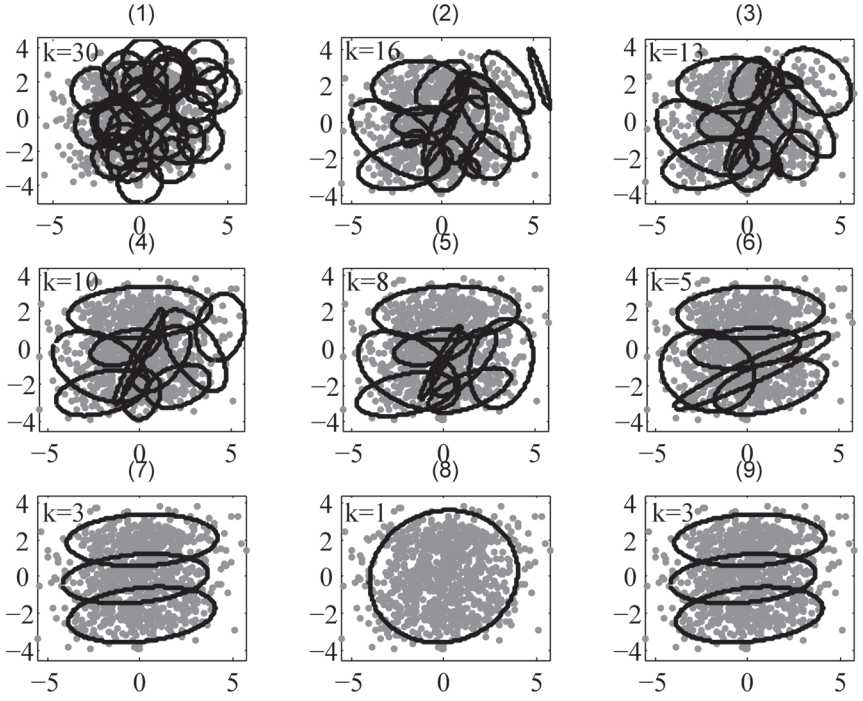

Figure 5. The demonstration of ULFMM using Gaussian mixture dataset in D2 example. Sub-figures (1) – (8) demonstrate the procedure of ULFMM. ULFMM starts from $K=30$ as shown sub-figure(1); ULFMM then merges clusters gradually to $K=1$ as shown in sub-figures(2) – (8) respectively. Sub-figure(9) is the final clustering result. Parameter setting: $k_{max}=30$.
doi:10.1371/journal.pone.0094141.g005

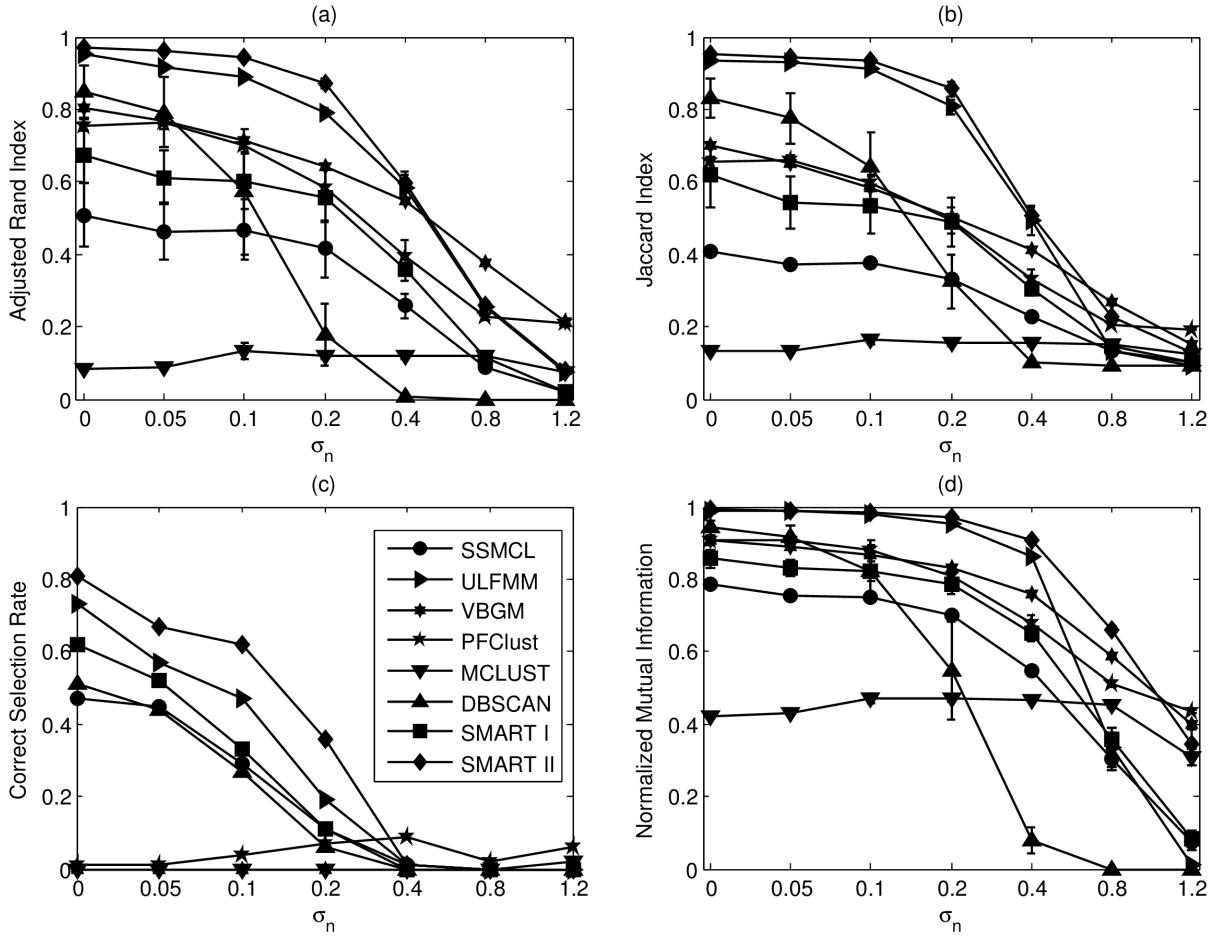

Figure 6. The errorbar charts of (a) ARI, (b) JI, (c) CSR, and (d) NMI for all compared algorithms in S1 datasets. The values of all four metrics are in the range of [0,1], where 1 is the optimal value and 0 is the worst one. The vertical axis in each sub-figure represents individual index and the horizontal axis is the standard deviation σ_n of the additive noise. SMART I and II are labelled with square and diamond markers respectively. SSMCL is labelled with circle marker, ULFMM is labelled with right-arrowed triangle marker, VGBM is labelled with by hexagon marker, PFClust is labelled with pentagon marker, MCLUST is labelled with down-arrowed triangle marker, and DBSCAN is labelled with up-arrowed triangle marker. For SMART I, $T_{chs}=20$; for SMART I and II, $N_m=5$. For ULFMM, SSMCL, and VBGM, $k_{max}=30$.
doi:10.1371/journal.pone.0094141.g006

been provided by Pramila et al. [65]. It is widely accepted that there are four phases in the cell cycle, namely, G1, S, G2 and M phases [66,67]. But there is no explicit knowledge about how many clusters should be in this dataset, so we cannot calculate CSR in this case. We obtain four clusters by using both SMART I and II, seven clusters by using ULFMM, eight clusters using SSMCL, three clusters using MFA with five factors, and five clusters using MCFA with six factors, as shown in Table 7. SMART II has the superior performance as in other experiments. We note that VBGM fails again in this experiment as it requires a dimension reduction of the data before clustering. We do not perform a reduction in data dimensions to obtain a fair comparison. To discern the effectiveness of the clusterings, we plot the histogram of the peak times of genes in each cluster for each algorithm, as depicted in Fig. 10, where the grey bar plot is the histogram of the 500 genes in the dataset. Fig. 10 (a) and (b) show that four clusters represent reasonably good clustering since there are only few small overlap regions between clusters. Fig. 10 (c) and (d) indicate that many clusters crowd and overlap in the region of 5% to 30%, especially in Fig. 10 (c), a clustering representing peaking at 20% superposes on another cluster, which spans over 10% to 30%. These overlapped clusters have to be one cluster. Fig. 10 (e) and (f) show that MFA and MCFA also give reasonably good clustering results judged by eye, however poorer than SMART II in the numerical metrics. Fig. 10 (g) and (h) show the distribution of peak times of genes based on the clustering results of MCLUST and PFClust, respectively. MCLUST has a very similar performance to MFA. The partition provided by PFClust has a cluster (labelled by brown circle) overlapping with other clusters. The numerical metrics consistently indicate that PFClust performs poorly in the R2 dataset. Since DBSCAN and VBGM do not provide a reasonable result, we do not depict it in Fig. 10. The results reveal that the SMART algorithms, especially, SMART II, provide a better representation than other algorithms.

We also compare the running time of the clustering algorithms for two real datasets in Table 8, where the algorithms implemented with MATLAB are listed in the upper section and the algorithms implemented with other platforms are in the lower section. For the sake of a fair comparison, we consider the running time of single run as the time consumed to find both best number of clusters and best partition, rather that the time only for clustering with one given number of clusters. The computer on which we conducted the experiments is equipped with Intel Core i7-3770 CPU 3.40 GHz and 8 GB RAM. According to the Table,

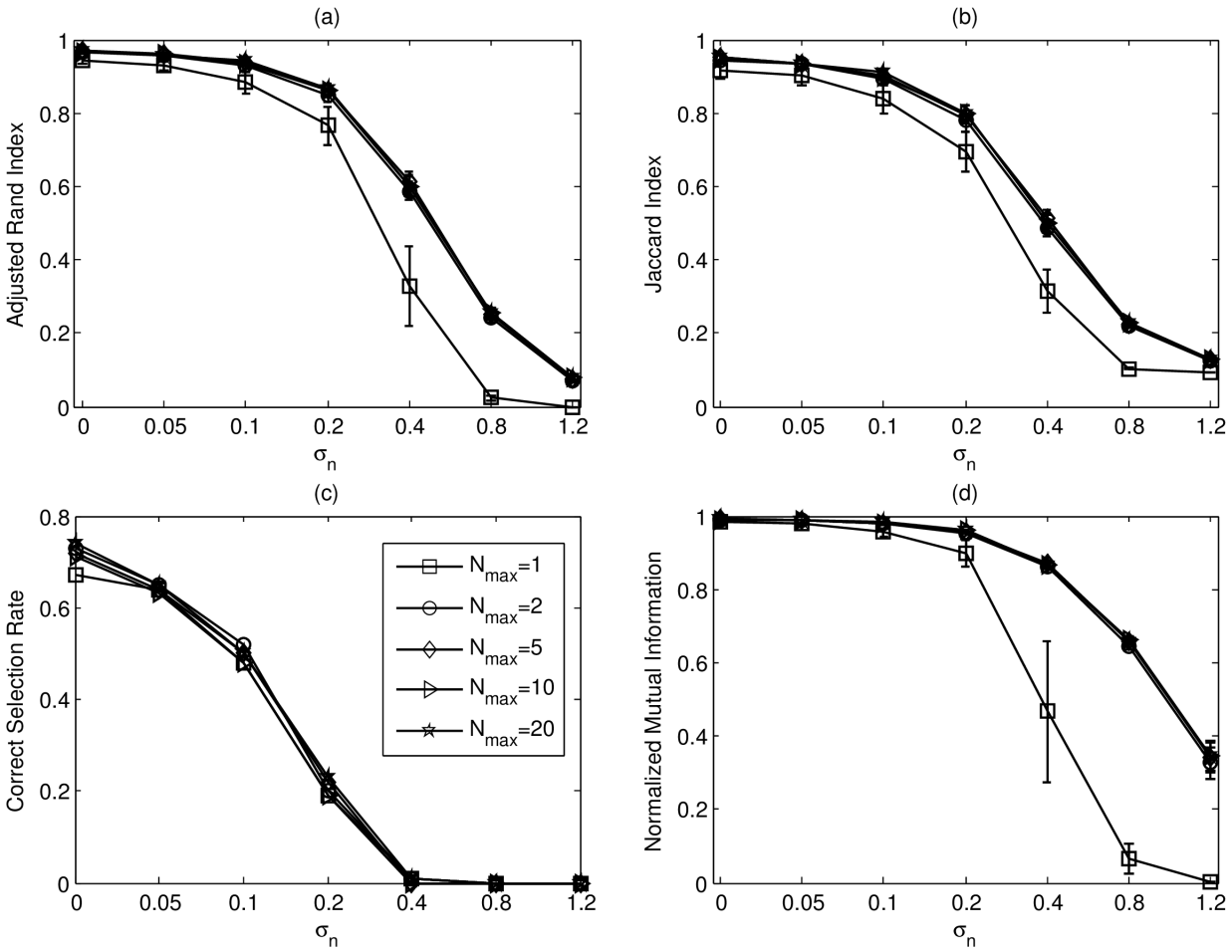

Figure 7. The errorbar charts of (a) ARI, (b) JI, (c) CSR, and (d) NMI for SMART II with different N_{max} values in S1 datasets. The vertical axis in each sub-figure represents individual index and the horizontal axis is the standard deviation σ_n of the additive noise. The line with square markers denotes $N_{max}=1$;The line with circle markers denotes $N_{max}=2$;The line with diamond markers denotes $N_{max}=5$;The line with triangle markers denotes $N_{max}=10$;The line with pentagon markers denotes $N_{max}=20$.
doi:10.1371/journal.pone.0094141.g007

SMART II consumed the least running time in both datasets. SMART I is faster that its counterpart algorithm SSMCL, but slower than ULFMM and VBGM. MFA and MCFA are time-consuming because they have to exhaustively search over both a range of number of clusters (K) and a range of number of factors (q), with a number of initial starts. The algorithms using other platforms, namely DBSCAN, MCLUST, and PFClust, also take longer time than the SMART algorithms to finish the same pieces of work.

Methods

SMART I

Here, we present the implementation of SMART I where OPTOC competitive learning is employed as the splitting and learning algorithm [48,49], cohesion is employed as merging criterion [54], and MML is employed as clustering selection criterion. The details of how these techniques work together is also presented.

OPTOC Competitive Learning. OPTOC competitive learning paradigm was firstly proposed in [48]. In SMART I, OPTOC competitive learning is employed to deal with Task 2. Given each prototype $\vec{\boldsymbol{P}}_k$, the key technique is that an online learning vector, asymptotic property vector (APV) $\vec{\boldsymbol{A}}_k$ is assigned to guide the learning of this prototype. For simplicity, $\vec{\boldsymbol{A}}_k$ represents the APV for prototype $\vec{\boldsymbol{P}}_k$ and $n_{\vec{\boldsymbol{A}}_k}$ denotes the learning counter (winning counter) of $\vec{\boldsymbol{A}}_k$. As necessary condition of OPTOC mechanism, $\vec{\boldsymbol{A}}_k$ is required to initialize at a random location, which is far from its associated prototype $\vec{\boldsymbol{P}}_k$ and $n_{\vec{\boldsymbol{A}}_k}$ is initially zero. Taking the input pattern $\mathbf{x}_i$ as a neighbour if it satisfies the condition $\langle \vec{\boldsymbol{P}}_k, \mathbf{x}_i \rangle \leq \langle \vec{\boldsymbol{P}}_k, \vec{\boldsymbol{A}}_k \rangle$, where $\langle \cdot, \cdot \rangle$ is the inner product operator. To implement the OPTOC paradigm, $\vec{\boldsymbol{A}}_k$ is updated online to construct a dynamic neighbourhood of $\vec{\boldsymbol{P}}_k$. The patterns "outside" of the dynamic neighbourhood will contribute less to the learning of $\vec{\boldsymbol{P}}_k$ as compared to those "inside" patterns.

In addition to the APV, there is another auxiliary vector, called distant property vector (DPV) $\vec{\boldsymbol{R}}_k$, assisting the cluster, which contains more than one prototype, to split. Let $n_{\vec{\boldsymbol{R}}_k}$ denote the learning counter for $\vec{\boldsymbol{R}}_k$, which is initialized to zero. $\vec{\boldsymbol{R}}_k$ will be updated to a distant location from $\vec{\boldsymbol{P}}_k$. The efficiency of splitting is improved by determining the update schedule of $\vec{\boldsymbol{R}}_k$ adaptively from the analysis of the feature space. Contrary to the APV $\vec{\boldsymbol{A}}_k$,

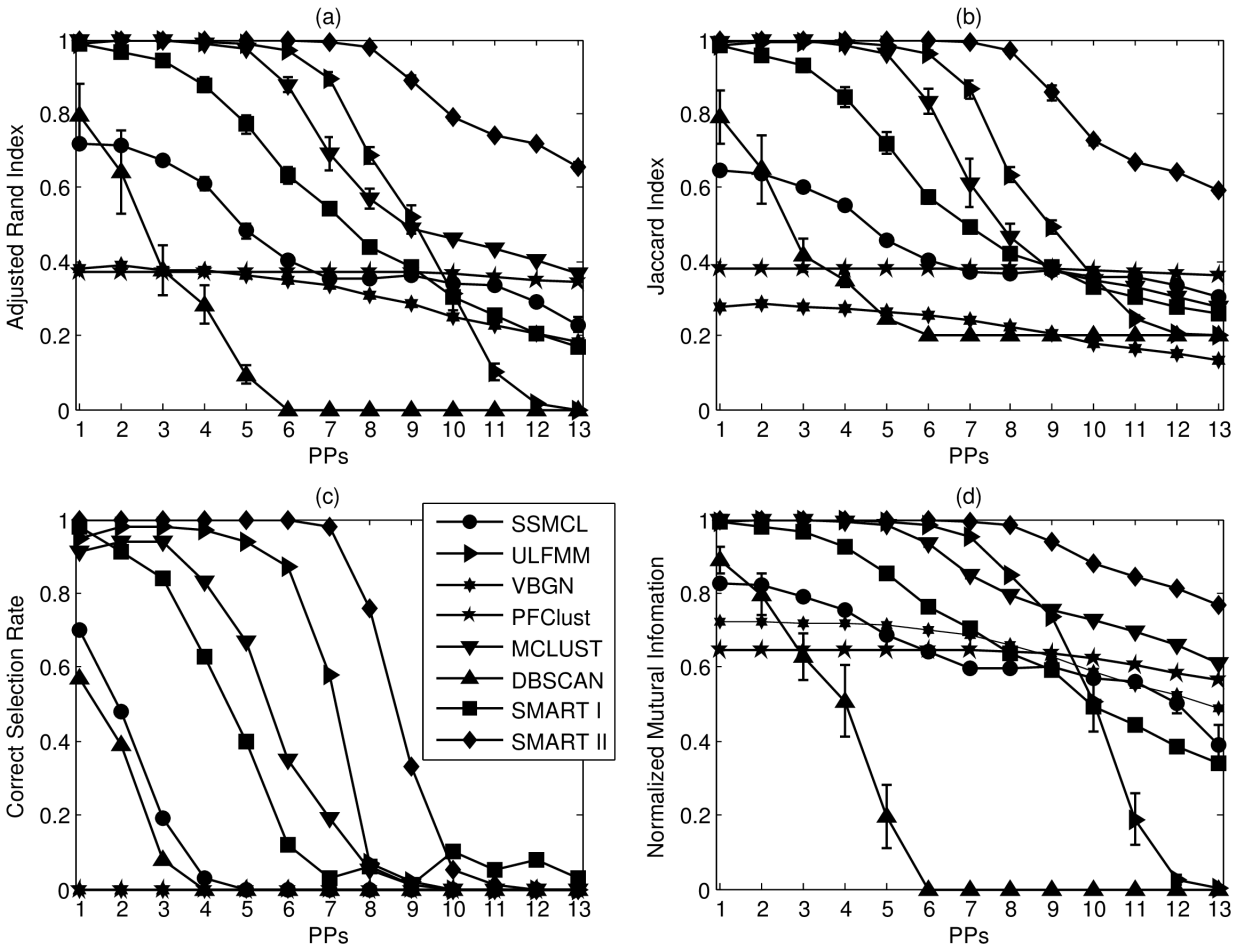

Figure 8. The errorbar charts of (a) ARI, (b) JI, (c) CSR, and (d) NMI for all compared algorithms in S2 datasets. The vertical axis in each sub-figure represents individual index and the horizontal axis is parameter pairs from PP1 to PP13, representing 13 noise levels from low to high. For SMART I, $T_{chs}=20$; for SMART I and II, $N_m=5$. For ULFMM, SSMCL, and VBGM, $k_{max}=30$.
doi:10.1371/journal.pone.0094141.g008

the DPV $\vec{\boldsymbol{R}}_k$ always tries to move away from $\vec{\boldsymbol{P}}_k$. Readers may refer to [48,49] for the details of updating $\vec{\boldsymbol{P}}_k$, $\vec{\boldsymbol{A}}_k$ and $\vec{\boldsymbol{R}}_k$.

Original OPTOC claims that the prototype converges if $\langle\vec{\boldsymbol{P}}_k,\vec{\boldsymbol{A}}_k\rangle<\epsilon$. However, ϵ is difficult to determine because it is data related. In our case, we define that the prototype $\vec{\boldsymbol{P}}_k$ converges if it satisfies

$$\left|1-\frac{\langle\vec{\boldsymbol{P}}_k,\vec{\boldsymbol{A}}_k\rangle}{\langle\vec{\boldsymbol{P}}_k^*,\vec{\boldsymbol{A}}_k^*\rangle}\right|<\epsilon', \tag{2}$$

where ϵ' is a positive constant smaller than one. It is worth noting that ϵ' is a relative number and is data-independent. Normally, smaller ϵ' leads to longer learning; while larger ϵ' leads to poorer performance. The suggested range of ϵ' is [0.001,0.005]. In our experiments, ϵ' is set to 0.005.

Cohesion. In [54], a similarity measure, namely cohesion, was proposed. The cohesion metrics is used for Task 2 in SMART I. It was defined as follows:

$$\mathrm{chs}(\mathcal{C}_k,\mathcal{C}_l)=\frac{\sum_{\boldsymbol{x}\in\mathcal{C}_k,\mathcal{C}_l}\mathrm{join}(\boldsymbol{x},\mathcal{C}_k,\mathcal{C}_l)}{|\mathcal{C}_k|+|\mathcal{C}_l|}, \tag{3}$$

where $\mathcal{C}_k$ is the cluster with the centroid $\boldsymbol{C}_k$, $|\mathcal{C}_k|$ is the size of the cluster of $\mathcal{C}_k$. $\mathrm{join}(\boldsymbol{x},\mathcal{C}_k,\mathcal{C}_l)$ defines the similarity of the two clusters referring to the existence of an object $\boldsymbol{x}$, which is defined as

$$\mathrm{join}(\boldsymbol{x},\mathcal{C}_k,\mathcal{C}_l)=\min(f_k(\boldsymbol{x}),f_l(\boldsymbol{x})), \tag{4}$$

where $f_k(\boldsymbol{x})$ and $f_l(\boldsymbol{x})$ are the probability density function (pdf) of the distributions in clusters $\mathcal{C}_k$ and $\mathcal{C}_l$. In our case we assume that an object in each cluster follows a multivariate normal distribution.

Minimum Message Length. Although there are a lot of model order selection algorithms and validity indices, we choose MML [30,31,50] for Task 4 in this work (both SMART I and SMART II) to avoid losing our focus by comparing different selection algorithms. MML is one of the minimum encoding length criteria, like the minimum description length (MDL) [29,56], and is used as the clustering selection algorithm. The rationale behind minimum encoding length criteria is that if one can build a short code for any given data, it implies that the code is a good model for fitting data. The shortest code length for set $\boldsymbol{\mathcal{X}}$ is $-\log p(\boldsymbol{\mathcal{X}}|\boldsymbol{\theta})$), where $\boldsymbol{\theta}$ contains the means $\boldsymbol{\mu}$ and the covariance matrices $\boldsymbol{\Psi}$. If $p(\boldsymbol{\mathcal{X}}|\boldsymbol{\theta})$ is fully known to both the transmitter and receiver, they can both build the same code and communication can proceed. However, if $\boldsymbol{\theta}$ is *a priori* unknown, the transmitter has to start by estimating and transmitting θ. This leads to a two-part

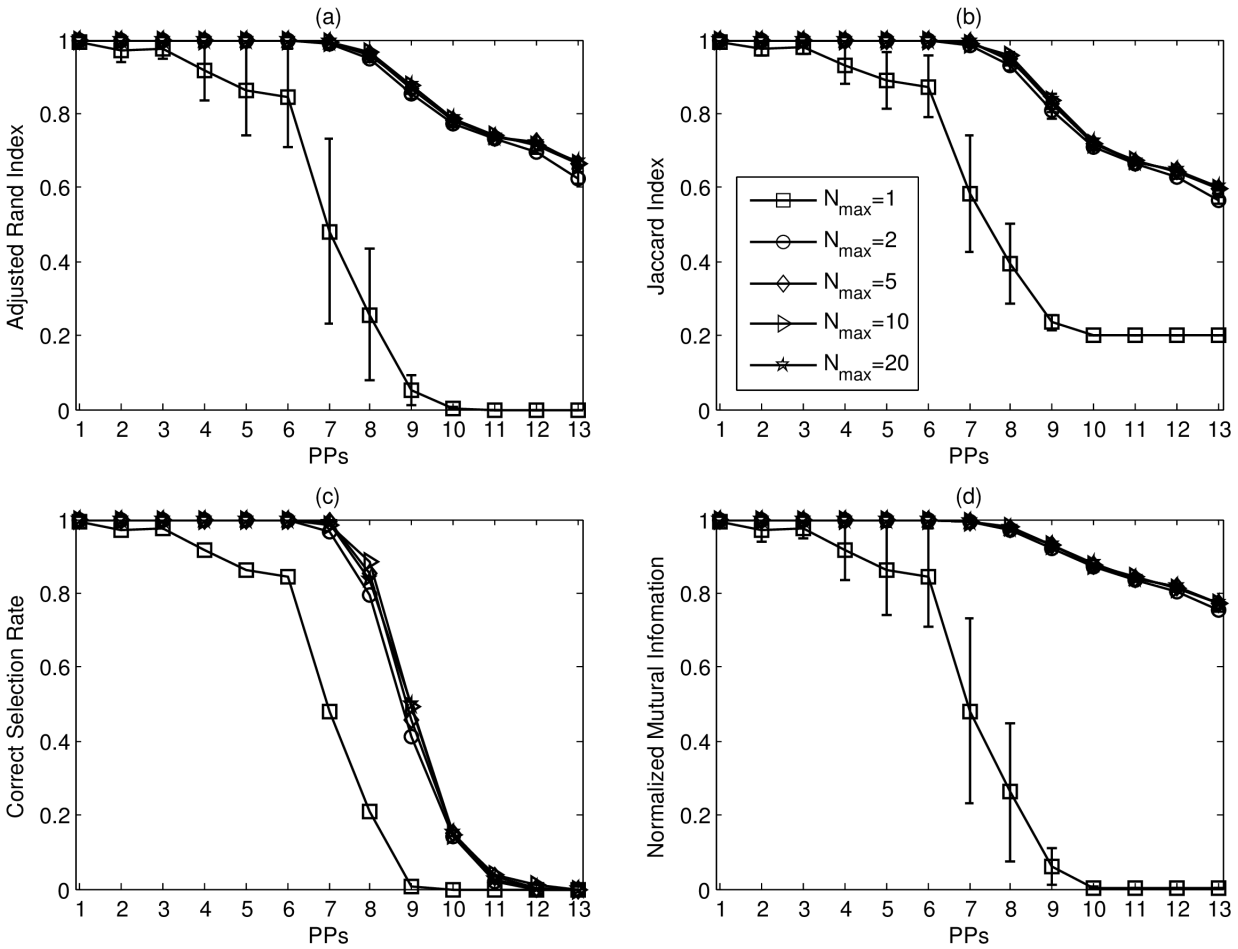

Figure 9. The errorbar charts of (a) ARI, (b) JI, (c) CSR, and (d) NMI for SMART II with different N_{max} values in S2 datasets. The vertical axis in each sub-figure represents individual index and the horizontal axis is parameter pairs from PP1 to PP13, representing 13 noise levels from low to high.
doi:10.1371/journal.pone.0094141.g009

message, whose total length is given by

$$\text{Length}(\boldsymbol{\theta},\boldsymbol{\mathcal{X}}) = \text{Length}(\boldsymbol{\theta}) + \text{Length}(\boldsymbol{\mathcal{X}}|\boldsymbol{\theta}). \quad (5)$$

All minimum encoding length criteria state that the parameter estimate is the one minimizing Length($\boldsymbol{\theta}$, $\boldsymbol{\mathcal{X}}$). The criterion was derived to the following form [50]

$$\text{Length}(\boldsymbol{\theta},\boldsymbol{\mathcal{X}}) = \frac{N_p}{2}\sum_{k=1}^{K}\log\alpha_k + \frac{N_p+1}{2}K\log N$$

$$-\log p(\boldsymbol{\mathcal{X}}|\boldsymbol{\theta}) + C, \quad (6)$$

where N_p is the number of parameters which is required in each component, $\{\alpha_k, 1 \le k \le K\}$ is the mixing probability of the k-th component with the constraint $\sum_{k=1}^{K}\alpha_k = 1$, and $C = (N_p+1)K(1-\log 12)/2$ is a constant. Note that the components with zero-probability in α_k have been eliminated and K is the number of non-zero-probability components.

SMART I Implementation. Here, we integrate these techniques into our SMART framework. The pseudo-code for SMART I is presented in Table 9.

Normally, Task 1 in SMART can be done by any initialization algorithms, either random or deterministic, like the KA algorithm [44]. In SMART I implementation presented here, a simple random initialization is used. The first prototype $\vec{\boldsymbol{P}}_1$ is randomly selected, the APV $\vec{\boldsymbol{A}}_1$ is the farthest object away from $\vec{\boldsymbol{P}}_1$, and the DPV $\vec{\boldsymbol{R}}_1$ is initialized as $\vec{\boldsymbol{P}}_1$. From then on, the SWM process starts. Learning with the OPTOC paradigm drags the prototype to its neighbour, which is "inside" the range of APV, and also drags the APV towards the prototype. Task 2 will not finish until every prototype converges. Since OPTOC is an online learning algorithm, systematic errors may be introduced by the order in which data is fed into the algorithm. Thus, every time OPTOC starts, the order of input data is randomized.

Once the prototypes converge, Task 3 commences. The pairwise cohesions are calculated to measure the distance between the prototype clusters. A criterion is set to guide the merging process, stating that if the maximum of the cohesions is T_{chs} times more than the majority of the cohesions, it reveals that the pair of two prototypes with this maximal cohesion are close enough to merge. The merging process continues until no further merge occurs. A merging counter records the number of merges. After

Table 5. Performance comparison of many metrics, including CSR, $\hat{K}$, MML, CH, SI for all algorithms in Leukemia dataset.

Algorithms	$\hat{K}(q)$	CSR	MML	CH	SI
MFA	3 (7)	/	4.23E4	6.42	0.35
MCFA	3 (4)	/	4.22E4	6.48	0.35
SSMCL	1 ± 0	0.0%	/	/	/
ULFMM	3.23 ± 0.54	69.4%	$3.91E4\pm2.07E2$	5.96 ± 0.89	0.32 ± 0.06
VBGM	30 ± 0	0.0%	$4.02E4\pm2.27E3$	0.78 ± 0.02	0.048 ± 0.013
DBSCAN	1 ± 0	0.0%	/	/	/
MCLUST	2 ± 0	0.0%	$4.27E4\pm0.0$	6.73 ± 0.0	0.36 ± 0.0
PFClust	4 ± 0	0.0%	$4.31E4\pm2.91$	$3.73\pm4.3E-3$	$0.21\pm2.51E-4$
SMART I	2.99 ± 0.13	99.0%	$3.89E4\pm1.62E2$	6.49 ± 0.3	0.36 ± 0.02
SMART II	**3 ± 0**	**100%**	**2.9E4±8.37E-3**	**6.75±5.64E-5**	**0.36±5.81E-8**

doi:10.1371/journal.pone.0094141.t005

the merging process finishes, the clustering is recorded as the candidate to output. If the merging counter exceeds the maximum number of merges N_m, the SWM process is terminated automatically; otherwise, it goes to Task 2 and continues splitting. Once the SWM process finishes, all the candidates are fed into the MML algorithm, which is associated with Task 4, to calculate $\text{Length}(\boldsymbol{\theta},\boldsymbol{\mathcal{X}})$. The final clustering results is the one, which minimizes $\text{Length}(\boldsymbol{\theta},\boldsymbol{\mathcal{X}})$.

Note that there are two parameters T_{chs} and N_{max} that have to be set in SMART, but they are neutral, i.e., T_{chs} is a relative number rather than absolute one, which is a data-independent value; the reason for setting N_{max} is that normally merging occurs frequently after the natural clustering has been reached. In our experiments, T_{chs} is set to 20 and N_{max} is set to 5. This is the key advantage over those over-clustering-then-merge algorithm, like SSMCL. The critical problem of SSMCL is that if the k_{max} is set too large, some prototypes have possibilities of being trapped in the low density area and difficult to converge.

SMART II

Here, we present the principal of SMART II, where the finite mixture model (FMM) is employed and the key technique is modified CEM^2 [50]. Since the FMM and the EM algorithm are very well-known topics, we will not address their details here and readers may refer to [57,58]. Since the conventional EM algorithm for mixture model has many drawbacks, e.g., it is sensitive to initialization and it is a local greedy method that may be trapped into local minima, the CEM^2 was proposed in [55] and modified in [50]. The greatest advantage of modified CEM^2 is that the weaker component may naturally be excluded in the iterative process, which gives the stronger ones a better chance of survival. From the merging point of view, it is a merging process combined with learning.

CEM^2 and Its Modification

Clustering dataset $\boldsymbol{\mathcal{X}}$, which follows a K-component finite mixture distribution, becomes the discovery of the missing labels $\boldsymbol{\mathcal{Z}}=\{z_1,...,z_N\}$ associated with the N data objects. Unlike conventional EM algorithm, CEM^2 updates the model parameters $\{\boldsymbol{\theta}_k|1\le k\le K\}$ and the probabilities of components $\{\alpha_k|1\le k\le K\}$ sequentially, rather than simultaneously. In CEM^2, the estimation is also two-step process, but in each iteration, only one component has the opportunity to update its parameters. For the j-component, it alternates the steps:

- **CEM^2** ***E-step***: Compute the conditional expectation $\boldsymbol{\Gamma}=\{\gamma_{k,i}|k=1,...,K;i=1,...,N\}$ of the missing labels $\boldsymbol{\mathcal{Z}}$ for $i=1,...,N$ and $k=1,...,K$,

Table 6. Performance comparison of SMART I and II with variable values of N_{max}.

		$N_{max}=5$	$N_{max}=10$	$N_{max}=20$
SMART I	MML	$3.89E4\pm1.62E2$	$4.00E4\pm1.52E2$	$4.00E4\pm2.01E2$
	CSR	99%	98.4%	98.4%
	$\hat{K}$	2.99 ± 0.13	2.98 ± 0.15	2.98 ± 0.15
	CH	6.49 ± 0.3	6.49 ± 0.13	6.49 ± 0.12
	SI	$0.36\pm2E-2$	$0.35\pm9.8E-3$	$0.35\pm1.2E-2$
SMART II	MML	$2.9E4\pm8.37E-4$	$3.27E\pm1.97E-3$	$3.26\pm1.76E-3$
	CSR	100%	100%	100%
	$\hat{K}$	3 ± 0	3 ± 0	3 ± 0
	CH	$6.75\pm5.64E-5$	$6.55\pm8.37E-5$	$6.55\pm6.11E-5$
	SI	$0.36\pm5.81E-8$	$0.36\pm5.83E-8$	$0.36\pm5.83E-8$

doi:10.1371/journal.pone.0094141.t006

Table 7. Performance comparison of many metrics, including $\hat{K}$, MML, CH, SI for all algorithms in yeast cell cycle dataset.

Algorithms	$\hat{K}(q)$	MML	CH	SI
MFA	3 (5)	1.36E4	6.68	0.37
MCFA	5 (6)	1.30E4	6.49	0.37
SSMCL	8	2.11E4	3.82	0.14
ULFMM	7	1.23E4	6.03	0.38
VBGM	20	3.97E4	1.98	0.17
DBSCAN	1	/	/	/
MCLUST	3	1.394	6.46	0.38
PFClust	6	1.24E4	3.94	0.32
SMART I	4	1.26E4	6.27	0.37
SMART II	4	**1.16E4**	**6.86**	**0.39**

doi:10.1371/journal.pone.0094141.t007

$$\gamma_{k,i} \equiv E[\hat{z}_{k,i}|\mathcal{X},\hat{\boldsymbol{\theta}}] = \frac{\hat{\alpha}_k p(\boldsymbol{x}_i|\hat{\boldsymbol{\theta}}_k)}{\sum_{l=1}^{K} \hat{\alpha}_l p(\boldsymbol{x}_i|\hat{\boldsymbol{\theta}}_l)}. \quad (7)$$

- **CEM^2 *M-step***: Set

$$\hat{\alpha}_j^* = \frac{\sum_{i=1}^{N} \gamma_{j,i}}{\sum_{l=1}^{K}\sum_{l=1}^{N} \gamma_{l,i}}, \quad (8)$$

$$\hat{\boldsymbol{\theta}}_j^* = \arg\max_{\boldsymbol{\theta}_j} \{\log p(\mathcal{X}|\hat{\boldsymbol{\theta}})\} \quad (9)$$

For $l \neq j$, $\hat{\alpha}_l^* = \hat{\alpha}_l$ and $\hat{\boldsymbol{\theta}}_l^* = \hat{\boldsymbol{\theta}}_l$.

In [50], the adoption of Dirichlet-type prior for α_ks results a new M-step

$$\hat{\alpha}_k^* = \frac{\max\left\{0, \sum_{i=1}^{N} \gamma_{k,i} - \frac{N_p}{2}\right\}}{\sum_{i=1}^{K}\left\{0, \sum_{i=1}^{N} \gamma_{l,i} - \frac{N_p}{2}\right\}}, \text{ for } k = 1,2,...,K. \quad (10)$$

The corresponding components $\hat{\boldsymbol{\theta}}_k$s with $\hat{\alpha}_k^* = 0$ is eliminated and become irrelevant. This component annihilation can be also explained in an estimation theoretic point of view as that the estimates are not accurate unless enough samples are involved. Those estimates without enough samples are dismissed and in turn others have more chances to survive. Modified CEM^2 can fulfil learning and merging, which are associated with Tasks 2 (only learning part) and 3, respectively, in SMART II.

SMART II Implementation

Compared with SMART I, SMART II is easier to implement since modified CEM^2 can do both that are learning and merging. In addition to the learning and merging techniques, there are two configurations different from SMART I. The first is that in SMART II, we initially start with $K = 2$ because $K = 1$ does not need learning, but $K = 1$ is still included in the candidate list for selection in the output. The second is that the splitting process cannot be done by modified CEM^2 and has to be specified. Once all components converge and all zero-probability components are discounted, a new component will be injected into the framework. This new component is initialized deterministically by using the farthest object away from the closet component among all the components as the mean and averaged covariance matrix of all components' covariance matrices, as given by

$$\mu_{K+1} = \arg\max_{\boldsymbol{x} \in \mathcal{X}} \{\min_{1 \leq k \leq K} \mathcal{D}(\boldsymbol{x}, \mu_k)\}, \quad (11)$$

$$\boldsymbol{\Psi}_{K+1} = \frac{1}{K}\sum_{k=1}^{K} \{\boldsymbol{\Psi}_k\}, \quad (12)$$

where $\mathcal{D}(\cdot)$ is a distance metric, and then the clustering splits $K = (K+1)$. The pseudo-code for SMART II is in Table 10. The stage for recoding the candidate clustering is after all current components converges and all merges finish and before the splitting for new component starts.

Discussion

We have developed a splitting-while-merging (SWM) clustering framework, named splitting-merging awareness tactics (SMART). The framework employs a SWM process and intrinsically integrates many clustering techniques. SMART has the ability to split and merge the clusters automatically during the process. Once the stop criterion is met, the SWM process terminates and the optimal clustering result is selected as final outcome by applying the selection criterion.

Although many recent algorithms have been proposed to achieve automated clustering, e.g. SSCL [48], ULFMM [50], SSMCL [49], PFClust [53], and VBGM [51,52], there are some issues that limit their practical use. For ULFMM, SSMCL, and VBGM, in spite of the fact that they do not require the exact value of K, they require the range of K, i.e. k_{max}, which is also not available sometimes. For PFClust, it needs a good approximation to the distribution of mean intra-cluster similarities, and it requires a large number of randomisation which is time-consuming. The main property of SMART is that it does not require any parameters dependent on respective datasets or *a priori* knowledge about the datasets, particularly, either the number of clusters or the possible range of this number.

Algorithms

Two SMART algorithms have been implemented with two distinct clustering paradigms: competitive learning for SMART I and learning with finite mixture model for SMART II. Competitive learning is a good candidate technique for on-line learning applications [49]. The selection criterion employs the minimum message length algorithm. It is worth noting that the components in the framework, e.g. the splitting, merging algorithms or the selection criterion, can be replaced by more powerful algorithms in the future, but the whole framework remains unchanged. We summarised the categorization of existing self-splitting-merging algorithms and our two SMART algorithms in Table 1. All existing self-splitting-merging algorithms employ the STM strategy with different clustering paradigms; instead our SMART algo-

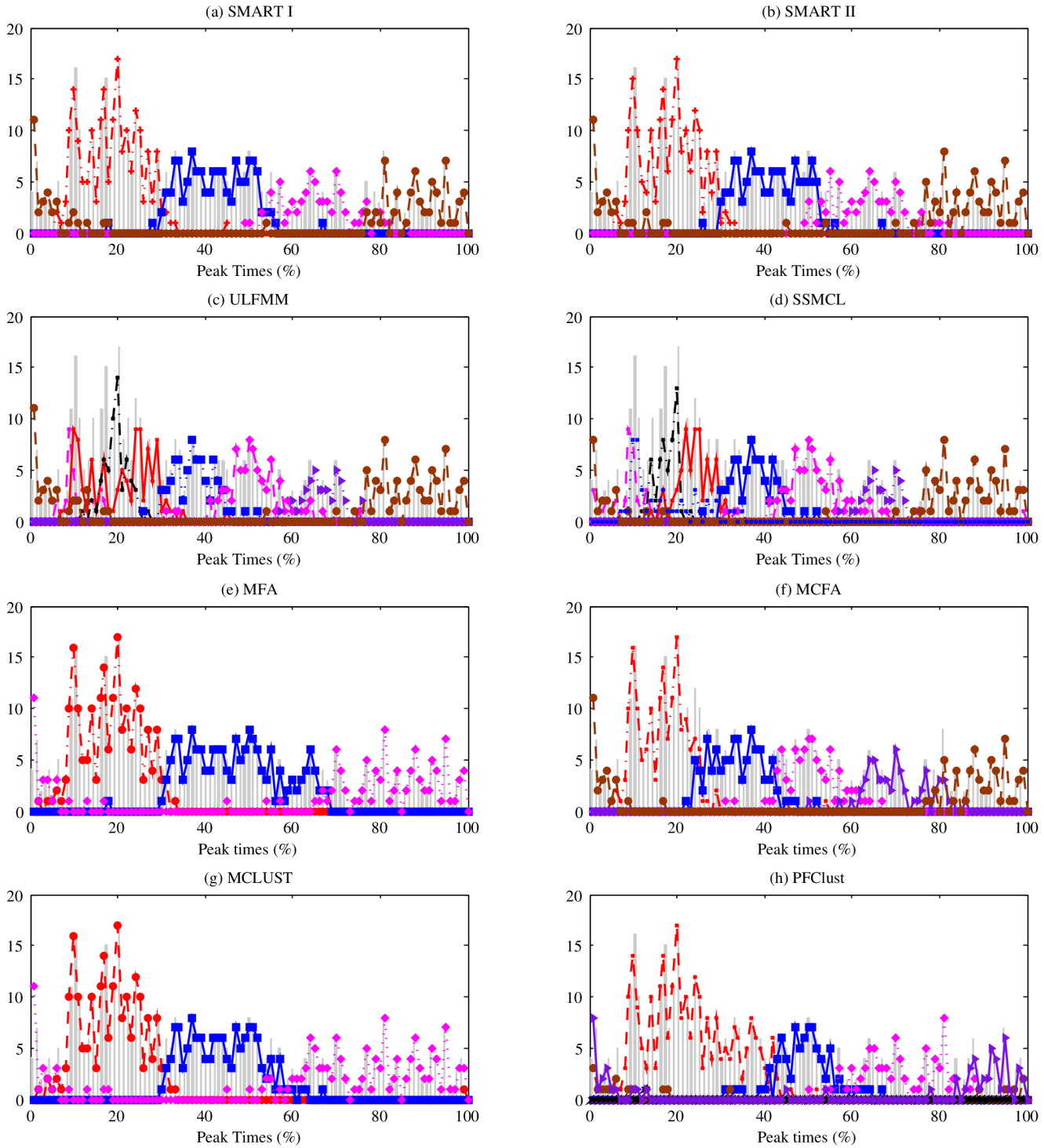

Figure 10. Histogram of the peak times of genes in each cluster for each algorithm in Yeast cell cycle α-38 dataset. (a) SMART I, $T_{chs}=20$ and $N_m=5$, $K=4$ (b) SMART II, $N_m=5$, $K=4$, (c) ULFMM,$k_{max}=30$, $K=7$, (d) SSMCL,$k_{max}=30$, $K=8$, (e) MFA, $q=5$, $K=3$, (f) MCFA, $q=6$, $K=5$, (g) MCLUST, $K=3$, (h) PFClust. Sub-figures (a) and (b) show that four clusters represent reasonably good clustering since there are only few small overlap regions between clusters. Sub-figures (c) and (d) indicate that many clusters crowd and overlap in the region of 5% to 30%, especially in Sub-figure (c), a clustering representing peaking at 20% superposes on another cluster, which spans over 10% to 30%. These overlapped clusters have to be one cluster. Sub-figures (e) and (f) show that MFA and MCFA also give reasonably good clustering results judged by eye, however poorer than SMART II in the numerical metrics. Sub-figures (g) and (h) show the distribution of the peak times of genes based on the clustering results of MCLUST and PFClust, respectively.
doi:10.1371/journal.pone.0094141.g010

Table 8. Comparison of running time (seconds) of the algorithms implemented in MATLAB (upper section) and other platforms (lower section) for two real datasets respectively.

Algorithms	R1 ($N=999, M=38$)	R2 ($N=500, M=25$)
MFA (MATLAB)	2.64E3	1.01E3
MCFA (MATLAB)	1.8E3	1.19E3
SSMCL (MATLAB)	43.68	7.18
ULFMM (MATLAB)	0.5	0.38
VBGM (MATLAB)	2.24	1.26
SMART I (MATLAB)	6.4	1.37
SMART II (MATLAB)	**0.47**	**0.37**
DBSCAN (R)	7.44	1.41
MCLUST (R)	165.10	13.44
PFClust (Java)	111.11	35.88

doi:10.1371/journal.pone.0094141.t008

rithms employ the SWM strategy. Both algorithms were detailed and tested using demonstration datasets as well as simulated gene expression datasets. We also noted that SMART can be implemented with other clustering paradigms without being restricted in the two techniques presented here. Such flexibility is apparently beneficial to extend SMART to many different applications.

Effectiveness of the SMART framework

Two demonstration examples illustrated the SWM process and showed the effectiveness of the proposed SMART framework. For different types of clustering techniques, the performance of the SMART algorithms varied. SMART I, for example, did not work well in the D2 dataset, since the CL-based algorithms are spherical. Two models [19,59], which simulates state-based gene expression data and time-sampled periodic gene expression data respectively, were employed to evaluate the clustering algorithms. In both types of simulated datasets, SMART-II offered remarkably better performance than others. Generally speaking, FMM-based algorithms performed better than CL-based algorithms in these two cases. Furthermore, two real microarray gene expression datasets [7,65] were studied using SMART. In these experiments, SMART-II also showed superior performance in many metrics. Particularly, SMART II has very small variations in these metrics, which means that it provides consistent results even though it is randomly initialized. Impressively, SMART I has significantly better performance than ULFMM in both real datasets. In the most cases except two demonstration examples, VBGM does not perform well as it is not suitable to directly cluster high dimensional datasets. One major issue of the STM framework, as one of bottom-up algorithms, is that it cannot produce a very accurate clustering result in some circumstances, since it makes clustering decisions based on local patterns without initially taking into account the global distribution. The SWM framework splits and merges the clusters in a top-down fashion to reach a global optimisation.

Table 9. The pseudo-code for SMART I.

Task 1: Initializing SMART with $K=1$

Randomly select $\vec{P}_1$ and find the farthest object as $\vec{A}_1$ and initialize $\vec{R}_1=\vec{P}_1$;

terminate $=0$;

while !terminate **do**

  Task 2: Use the OPTOC paradigm for the learning of prototype, and the

  splitting of the cluster with largest variance;

  if the prototype $\vec{P}_k$ does not converge **then**

    Go back to Task 2;

  end if

  Task 3: Calculate pairwise cohesions for all converged prototypes (3);

  if The maximum of cohesions is T_{chs} times larger than the median of cohesions

  then

    Merge the pair of cluster with the maximum cohesion;

    Go back to Task 3 to continue merging;

  end if

  The stage for recoding candidate clustering.

  if The number of merges is greater than or equal to N_m **then**

    terminate $=1$;

  end if

end while

Task 4: Calculate the length for every converged clustering, output the clustering

with the minimum length.

doi:10.1371/journal.pone.0094141.t009

Table 10. The pseudo-code for SMART II.

Task 1: Initializing SMART with $K=2$
Randomly initialize $\hat{\boldsymbol{\theta}}_k$ and $\hat{\alpha}_k$ for $k=1,2$;
terminate $=0$;
while !terminate **do**
Tasks 2 & 3: Use modified CEM^2 for the learning and merging based on (7) and (10).
if the prototype $\hat{\boldsymbol{\theta}}_k$ does not converge **then**
Go back to Tasks 2 & 3;
end if
The stage for recoding candidate clustering.
Splitting: Calculate the parameters for new components (11) and (12);
if The number of merges is greater than or equal to N_m
terminate $=1$;
end if
end while
Task 4: Calculate the length for every converged clustering, output the clustering with the minimum length.

doi:10.1371/journal.pone.0094141.t010

Summary

We have proposed a new clustering framework named SMART which possesses three outstanding properties: 1) by integrating many clustering techniques including clustering paradigm, clustering validation, and clustering measure, the proposed SMART framework does not require any parameters dependent on the respective dataset or *a priori* knowledge about the datasets; (2) the implementation of the SMART framework is flexible and extendible to different applications; (3) the SMART algorithms appears to produce more accurate clustering results than counterpart algorithms.

Future Work

In future work, we will derive new algorithms based on other clustering paradigms, which could be either more robust for general clustering purposes or more appropriate to some particular type of data. Additionally, SMART will be applied in consensus clustering [24,25], which can achieve consistency among different clustering results of same set of genes in different datasets. Since the critical issue of consensus clustering is the determination of the number of clusters, SMART can overcome this problem and produce different clustering results to many different datasets without specifying any parameters related to respective datasets. Combining these clustering results will reveal consistently co-expressed genes, which have higher possibility to be co-regulated. This can be beneficial in either gene discovery or gene regulatory networks research.

Acknowledgments

This article summarises independent research funded by the National Institute for Health Research (NIHR) under its Programme Grants for Applied Research Programme (Grant Reference Number RP-PG-0310-1004). The views expressed are those of the authors and not necessarily those of the NHS, the NIHR or the Department of Health. Prof. A. K. Nandi would like to thank TEKES for their award of the Finland Distinguished Professorship.

Author Contributions

Conceived and designed the experiments: RF AKN. Performed the experiments: RF AKN. Analyzed the data: RF AKN. Wrote the paper: RF DJR AKN.

References

1. Jain AK, Murty MN, Flynn PJ (1999) Data clustering: A review. ACM Computing Surveys 31: 3, 316–323.
2. Jain AK, Duin RPW, Mao J (2000) Statistical pattern recognition: A review. IEEE Transactions on Pattern Analysis and Machine Intelligence 22: 4–37.
3. Jiang D, Tang C, Zhang A (2004) Cluster analysis for gene expression data: A survey. IEEE Transactions on Knowledge and Data Engineering 16: 1370–1386.
4. Xu R, Wunsch D (2005) Survey of clustering algorithms. IEEE Transactions on Neural Networks 16: 645–78.
5. Xu R, Wunsch DC (2010) Clustering algorithms in biomedical research: a review. IEEE Reviews in Biomedical Engineering 3: 120–54.
6. Eisen MB, Spellman PT, Brown PO, Botstein D (1998) Cluster analysis and display of genome-wide expression patterns. Proc. Natl. Acad. Sci. 14863–14868.
7. Golub TR, Slonim DK, Tamayo P, Huard C, Gaasenbeek M, et al. (1999) Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. Science 286: 531–537.
8. Tamayo P, Slonim D, Mesirov J, Zhu Q, Kitareewan S, et al. (1999) Interpreting patterns of gene expression with self-organizing maps: methods and application to hematopoietic differentiation. Proc. Natl. Acad. Sci. 96: 2907–2912.
9. Dembele D, Kastner P (2003) Fuzzy c-means method for clustering microarray data. Bioinformatics 19: 973–980.
10. Qin J, Lewis DP, Noble WS (2003) Kernel hierarchical gene clustering from microarray expression data. Bioinformatics 19: 2097–2104.
11. Slonim N, Atwal GS, Tkačik G, Bialek W (2005) Information-based clustering. Proc. Natl. Acad. Sci., 102(51), 18297–18302.
12. Bandyopadhyay S, Mukhopadhyay A, Maulik U (2007) An improved algorithm for clustering gene expression data. Bioinformatics 23: 2859–2865.
13. Boly M, Perlbarg V, Marrelec G, Schabus M, Laureys S, et al. (2012) Hierarchical clustering of brain activity during human nonrapid eye movement sleep. Proc. Natl. Acad. Sci., 109(15), 5856–5861.
14. Girvan M, Newman MEJ (2002) Community structure in social and biological networks Proc. Natl. Acad. Sci., 99 (12), 7821–7826.
15. Newman MEJ (2006) Modularity and community structure in networks, Proc. Natl. Acad. Sci., 103:23, 8577–8582.
16. Fortunato S (2010) Community detection in graphs, Physics Reports, 486 (3-5), 75–174.
17. Yeung KY, Fraley C, Murua A, Raftery AE, Ruzzo WL (2001) Model-based clustering and data transformations for gene expression data. Bioinformatics 17: 977–987.

18. McLachlan GJ, Bean RW, Peel D (2002) A mixture model-based approach to the clustering of microarray expression data, Bioinformatics, 18:3, 413–422.
19. Thalamuthu A, Mukhopadhyay I, Zheng X, Tseng GC (2006) Evaluation and comparison of gene clustering methods in microarray analysis. Bioinformatics 22: 2405–2412.
20. Baek J, McLachlan GJ (2011) Mixtures of common t-factor analyzers for clustering high-dimensional microarray data. Bioinformatics 27, 1269–1276.
21. Baek J, McLachlan GJ, Flack L (2010) Mixtures of factor analyzers with common factor loadings: applications to the clustering and visualisation of high-dimensional data. IEEE Transactions on Pattern Analysis and Machine Intelligence 32, 1298–1309.
22. McNicholas PD, Murphy TB (2010) Model-based clustering of microarray expression data via latent Gaussian mixture models, Bioinformatics 26(21), 2705–2712.
23. Fa R, Nandi AK, Gong LY (2012) Clustering analysis for gene expression data: A methodological review. 5th International IEEE Symposium on Communications Control and Signal Processing (ISCCSP).
24. Abu-Jamous B, Fa R, Roberts DJ, Nandi AK (2013) Paradigm of Tunable Clustering Using Binarization of Consensus Partition Matrices (Bi-CoPaM) for Gene Discovery. PloS ONE 8:2, e56432. doi:10.1371/journal.pone.0056432
25. Abu-Jamous B, Fa R, Roberts DJ, Nandi AK, (2013) Yeast gene CMR1/YDL156W is consistently co-expressed with genes participating in DNA-metabolic processes in a variety of stringent clustering experiments. J R Soc Interface 2013 10: 20120990
26. Milligan GW, Cooper MC (1985) An examination of procedures for determining the number of clusters in a data set, Psychometrika, 50:2, 159–179.
27. Halkidi M, Batistakis Y, Vazirgiannis M (2002) Cluster validity methods: Part I, SIGMOD, 31:2, 40–45.
28. Halkidi M, Batistakis Y, Vazirgiannis M (2002) Cluster validity methods: Part II, SIGMOD, 31:3, 19–27.
29. Rissanen J (1978) Modeling by shortest data description. Automatica 14: 465–471.
30. Oliver JJ, Baxter RA, Wallace CS (1996) Unsupervised learning using MML. In: Proceeding of International Workshop Machine Learning. 364–372.
31. Wallace CS, Dowe DL (1999) Minimum message length and kolmogorov complexity. The Computer Journal 42: 270–283.
32. Fraley C, Raftery A (1998) How Many Clusters? Which Clustering Method? Answers Via Model-Based Cluster Analysis, Technical Report 329, Dept. Statistics, Univ. Washington, Seattle, WA.
33. Akaike H (1974) A new look at the statistical model identification. IEEE Transactions on Automatic Control, 19.6: 716–723.
34. Celeux G, Gilda S (1996) An entropy criterion for assessing the number of clusters in a mixture model. Journal of classification 13.2: 195–212.
35. Calinski T, Harabasz J (1974) A dendrite method for cluster analysis, Communications in Statistics - Theory and Methods, 3:1, 1–27.
36. Dunn JC (1973) A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters, J. Cybernetics, 3:3, 32–57.
37. Davies DL, Bouldin DW (1979) A Cluster Separation Measure, IEEE Transactions on Pattern Analysis and Machine Intelligence, 1: 2, 224–227.
38. Maulik U, Bandyopadhyay S (2002) Performance evaluation of some clustering algorithms and validity indices, IEEE Transactions on Pattern Analysis and Machine Intelligence, 24: 12, 1650–1654.
39. Rousseeuw PJ (1987) Silhouettes: A graphical aid to the interpretation and validation of cluster analysis, Journal of Computational and Applied Mathematics, 20, 53–65.
40. Lam BSY, Yan H (2007) Assessment of microarray data clustering results based on a new geometrical index for cluster validity, Soft Computing, 11:4, 341–348.
41. Salem SA, Nandi AK (2009) Development of assessment criteria for clustering algorithms, Pattern Analysis and Applications, 12:1, 79–98.
42. Fa R, Nandi AK (2011) Parametric validity index of clustering for microarray gene expression data, in IEEE Int. Workshop Machine Learning for Sig. Process. (MLSP).
43. Fa R, Nandi AK (2014) Noise Resistant Generalized Parametric Validity Index of Clustering for Gene Expression Data, IEEE Transaction on Computational Biology and Bioinformatics, *in press*.
44. Rousseeuw PJ, Kaufman L (1990) Finding groups in data: An introduction to cluster analysis, John Wiley & Sons, Oxford, UK.
45. Kohonen T (1990) The self-organizing maps, Proceedings of the IEEE, 78(9): 1464–1480.
46. Fritzke B (1994) Growing cell structuresa self-organizing network for unsupervised and supervised learning, Neural Networks 7: 1441–1460.
47. Fritzke B (1995) A growing neural gas network learns topologies, Advances in Neural Information Processing Systems 7: 625–632.
48. Zhang YJ, Liu ZQ (2002) Self-splitting competitive learning: A new on-line clustering paradigm, IEEE Transactions on Neural Networks 13: 369–380.
49. Wu S, Liew AC, Yan H, Yang M (2004) Cluster analysis of gene expression data based on self-splitting and merging competitive learning, IEEE Transactions on Information Technology in Biomedicine 8: 5–15.
50. Figueiredo MAT, Jain AK (2002) Unsupervised learning of finite mixture models, IEEE Transactions on Pattern Analysis and Machine Intelligence 24: 381–396.
51. Teschendorff AE, Wang YZ, Barbosa-Morais NL, Brenton JD, Caldas C (2005) A variational Bayesian mixture modelling framework for cluster analysis of gene-expression data, Bioinformatics 21(13): 3025–3033
52. Bishop CM, Nasrabadi NM (2006) Pattern recognition and machine learning, New York: springer, 2006.
53. Mavridis L, Nath N, Mitchell JB (2013) PFClust: a novel parameter free clustering algorithm, BMC bioinformatics,14(1):213
54. Lin CR, Chen MS (2005) Combining partitional and hierarchical algorithms for robust and efficient data clustering with cohesion self-merging, IEEE Transactions on Knowledge and Data Engineering 17: 145–159.
55. Celeux G, Chrétien S, Forbes F, Mkhadri A (1999) A component-wise EM algorithm for mixtures, Technical Report 4, INRIA Rhone-Alpes, France.
56. Lanterman AD (2001) Schwarz, wallace, and rissanen: Intertwining themes in theories of model selection, International Statistical Review 69: 185–212.
57. McLachlan G, Peel D (2000) Finite mixture models, Wiley-Interscience, New Jersey.
58. McLachlan GJ, Krishnan T (2007) The EM algorithm and extensions, Wiley-Interscience, New Jersey.
59. Zhao L, Prentice R, Breeden L (2001) Statistical modeling of large microarray data sets to identify stimulus-response profiles, Proc. Natl. Acad. Sci. 98: 5631–5636.
60. Ester M, Kriegel HP, Sander J, Xu X (1996).A density-based algorithm for discovering clusters in large spatial databases with noise, In KDD 96, pp. 226–231.
61. Rand WM (1971) Objective criteria for the evaluation of clustering methods, Journal of the American Statistical Association 66: 846–850.
62. Hubert L, Arabie P (1985) Comparing partitions, Journal of Classification 2: 193–218.
63. Monti S, Tamayo P, Mesirov J, Golub T (2003) Consensus clustering: a resampling-based method for class discovery and visualization of gene expression microarray data, Machine Learning 52: 91–118.
64. Fa R, Nandi AK (2011) Comparisons of validation criteria for clustering algorithms in microarray gene expression data analysis, In: The Second Int. Workshop on Genomic Sig. Proc. (GSP2011).
65. Pramila T, Wu W, Miles S, Noble WS, Breeden LL (2006) The forkhead transcription factor hcm1 regulates chromosome segregation genes and fills the s-phase gap in the transcriptional circuitry of the cell cycle, Genes & Development 20: 2266–2278.
66. Cho RJ, Campbell MJ, Winzeler EA, Steinmetz L, Conway A, et al. (1998) A genome-wide transcriptional analysis of the mitotic cell cycle, Molecular Cell, 2:65–73.
67. Spellman PT, Sherlock G, Zhang MQ, Iyer VR, Anders K, et al. (1998) Comprehensive Identification of Cell Cycle regulated Genes of the Yeast Saccharomyces cerevisiae by Microarray Hybridization, Molecular Biology of the Cell 9:3273–3297.