Proceedings of the 5th International Symposium on Communications, Control and Signal Processing,
ISCCSP 2012, Rome, Italy, 2-4 May 2012

CLUSTERING ANALYSIS FOR GENE EXPRESSION DATA: A METHODOLOGICAL REVIEW

Rui Fa[1], *Asoke K Nandi*[1,2] *and Li-Yun Gong*[1]

[1] Department of Electrical Engineering and Electronics, The University of Liverpool,
Brownlow Hill, Liverpool, L69 3GJ, UK
[2]Department of Mathematical Information Technology, University of Jyväskylä,
Jyväskylä, Finland.
Email: {r.fa, a.nandi, l.gong}@liverpool.ac.uk

ABSTRACT

Clustering is one of most useful tools for the microarray gene expression data analysis. Although there have been many reviews and surveys in the literature, many good and effective clustering ideas have not been collected in a systematic way for some reasons. In this paper, we review five clustering families representing five clustering concepts rather than five algorithms. We also review some clustering validations and collect a list of benchmark gene expression datasets.

Index Terms— Clustering algorithm, Clustering validation, Microarray gene expression data analysis.

1. INTRODUCTION

Microarray is a comparatively new technology and a chip-based high throughput technology to investigate the expression levels of thousands of genes simultaneously, compared with the traditional approach to genomic research. Expression data are collected via either experiments in a time series during a biological process or experiments of different tissue samples [1]. A gene expression dataset is organized in a expression matrix, where the rows represent the expression profiles or patterns of genes, and the columns represent the expression profiles of samples. Data preprocessing is required before any clustering analysis, because the original gene expression matrix contains noise, missing values and systematic variations. By analysing the gene expression across multiple experiments, co-regulated gene with similar biological functions and their interactions, or their characteristic of the states, diseases, or phenotypes represented by groups of samples may be discovered.

Clustering, also known as unsupervised learning, has been used for decades in many fields, such as image processing, data mining and artificial intelligence [2], and in recent years, has benefited microarray gene expression data analysis in genomic research [3]. The goal of the clustering analysis is to group individual objects or samples in a population within which the objects are more similar to each other than those in other clusters. Generally speaking, to study or design a clustering analysis for an application, one has to consider three issues: 1) the measurements of the dissimilarity (or similarity), 2) the clustering algorithms, 3) the clustering validations. There has been a rich literature on clustering analysis over the past decades and all these three issues have been comprehensively discussed in [2, 3]. Especially in [3], the literature of clustering analysis for gene expression data have been reviewed and discussed.

The project (Ref. NIHR-RP-PG-0310-1004-AN) is supported by National Institute for Health Research (NIHR), UK.

There were eight similarity and dissimilarity measures listed in [2], namely, *Minkowski distance*, *Euclidean distance*, *City-block distance*, *Sup distance*, *Mahalanobis distance*, *Pearson correlation*, *Point symmetry distance*, *Cosine similarity*, which have been widely used in various applications. In [3], *Euclidean distance* and *Pearson correlation* were claimed to be effective similarity measures for gene expression data. Furthermore, two additional measures, namely *Jackknife correlation* and *Spearman's rank-order correlation*, were discussed to cope with the situations of outliers and non-Gaussian distributions, respectively. In this paper, we will not discuss the dissimilarity and similarity measures but use the operators $\mathcal{D}(\cdot)$ for dissimilarity and $\mathcal{S}(\cdot)$ for similarity instead of a specific measure when we study clustering algorithms. Readers who are interested in more details are advised to refer to [2, 3] and the references therein.

In this paper, we focus our attention on reviewing and discussing five families of clustering algorithms, some clustering validations and some benchmark microarray gene expression datasets. As it has been a few years since those two comprehensive review papers [2, 3] were published, many new and effective algorithms have been proposed but were not reviewed. Especially, for gene expression data analysis, both the clustering algorithms and validations have grown beyond the horizon of [3]. This paper can be viewed as a complementary counterpart to make the literature review in this field somehow up-to-data. Excluding some popular clustering algorithms, say *k-means*, *k-medoids*, *hierarchical*, *model-based clustering etc.*, we will discuss five different families of clustering algorithms including *fuzzy clustering*, *kernel-based clustering*, *self-organizing clustering*, *self-splitting and merging clustering* and *ensemble clustering*. We will also review some clustering validation methods used for gene expression data analysis, which were not included in [3]. Furthermore, we will collect a list of benchmark gene expression datasets to which clustering analysis has commonly been applied.

The rest of the paper is organized as follows. In Sec. 2 we review five families of clustering algorithms, sequentially, *fuzzy clustering*, *kernel-based clustering*, *self-organizing clustering*, *self-splitting and merging clustering* and *ensemble clustering*. Sec. 3 reviews the clustering validations. The benchmark gene expression datasets are listed in Sec. 4. Finally, concluding remarks are made in Sec. 5.

2. CLUSTERING ALGORITHMS

The five clustering families, which are reviewed in this section, represent five different methodologies of clustering, though there may exist some commonalities in their implementation. Throughout the paper, we suppose that we are going to partition the gene expression dataset $\mathbf{X} = \{\boldsymbol{x}_n | 1 \leq n \leq N\}$, where $\boldsymbol{x}_n \in \mathbb{R}^{M\times 1}$ denotes

978-1-4673-0276-0/12/$31.00 ©2012 IEEE

Table 1. The procedure of FCM

STEP 1:
Initialize the centroids of clusters $\{\boldsymbol{c}_k|k=1,...,K\}$ randomly or based on some prior knowledge if available;
STEP 2:
Update the membership matrix $\boldsymbol{U}$ by
$u_{kn} = 1/\left[\sum_{k'=1}^{K}\left(\frac{\mathcal{D}(\boldsymbol{x}_n,\boldsymbol{c}_{k'})}{\mathcal{D}(\boldsymbol{x}_n,\boldsymbol{c}_k))}\right)^{2/(1-m)}\right], k=1,...,K$ and $n=1,...,N$.
STEP 3:
Update the centroids of clusters $\{\boldsymbol{c}_k|k=1,...,K\}$ by
$\boldsymbol{c}_k(t) = \frac{\left(\sum_{n=1}^{N} u_{k,n}^m \boldsymbol{x}_n\right)}{\left(\sum_{n=1}^{N} u_{k,n}^m\right)}$, for $k=1,...,K$.
STEP 4:
Repeat **STEP 2 and 3** until $\|\boldsymbol{C}(t)-\boldsymbol{C}(t-1)\| < \epsilon$, where ϵ is a small positive number.

the n-th gene, M is the number of samples (features or dimensions) and N is the number of genes. Consider that there are K clusters in a given dataset and each clustering algorithm provides a partition matrix $\boldsymbol{U}^{K\times N}$, where the entry $u_{k,n} \in [0,1]$ represents the membership coefficient of n-th gene in the k the cluster.

2.1. Fuzzy Clustering

Fuzzy clustering is a concept that relaxes the restriction of crisp clustering, which assigns every object exactly in one clusters, by characterizing the membership of each sample point in all the clusters with a membership function which ranges between zero and one. Additionally, the sum of the memberships for each sample point must be unity [4]. The properties of the partition matrix is mathematically expressed by

$$\begin{aligned}
&(a) \quad u_{k,n} \in [0,1], \quad 1 \le n \le N, 1 \le k \le K,\\
&(b) \quad \textstyle\sum_{k=1}^{K} u_{k,n} = 1, \quad 1 \le n \le N,\\
&(c) \quad 0 < \textstyle\sum_{n=1}^{N} u_{k,n} < N, \quad 1 \le k \le K.
\end{aligned}$$

An example of fuzzy clustering is the fuzzy c-means (FCM) [5], which is fuzzy counterpart of the k-means. FCM aims to minimize the cost function, which is mathematically expressed by

$$\mathcal{J}(\boldsymbol{U},\mathbf{X}) = \sum_{k=1}^{K}\sum_{n=1}^{N}(u_{k,n})^m \mathcal{D}(\boldsymbol{x}_n,\boldsymbol{c}_k), \tag{1}$$

where $m \in [1,\infty)$ is the fuzzification parameter. $\mathcal{D}(\boldsymbol{x}_n,\boldsymbol{c}_k)$ is the dissimilarity measure between the n-th gene, $\boldsymbol{x}_n$, and the centroid of the k-th cluster, $\boldsymbol{c}_k$. Similar to k-means, the cost function (1) can be minimised with an iterative procedure that updates the partition matrix and the centroids of clusters alternately. Since many genes may participate in more than one function in the biological process, the fuzzy clustering has obvious advantage to identify some genes co-regulating with more than one cluster of genes. Many applications of FCM and its enhancements and modifications for gene expression data have been developed [6–10]. The procedure of FCM is summarized in Table 1. There are also many other fuzzy clustering approaches, for example, fuzzy self-organizing map (FSOM) [11], fuzzy adaptive resonance theory (FART) [12] and fuzzy support vector machine (FSVM) [13].

2.2. Kernel-based Clustering

Kernel-based clustering, which shares similar idea with support vector machines (SVM), constructs a hyperplane to separate the patterns. These patterns are nonlinearly transformed from a set of nonlinearly separable patterns into a higher-dimensional feature space to be linear separable [14, 15]. At the core of the kernel-based clustering lies the difficulty of explicitly constructing the nonlinear mapping, $\Phi(\cdot)$, which is sometime infeasible; but now it can be overcome by a kernel trick. The kernel trick is a way of mapping patterns from a input space into a feature space without having to compute the mapping explicitly, in the hope that the patterns will gain meaningful linear structure in the feature space, mathematically expressed as

$$\boldsymbol{\kappa}(\boldsymbol{x}_i,\boldsymbol{x}_j) = \Phi(\boldsymbol{x}_i)^T\Phi(\boldsymbol{x}_j), \tag{2}$$

where $(\cdot)^T$ is the transpose operator. Thus, a straightforward way to transform the calculation of Euclidean distance in the feature space into the kernel version is to use the kernel trick as follows

$$\begin{aligned}
\mathcal{D}_E^{\kappa}(\Phi(\boldsymbol{x}_i),\Phi(\boldsymbol{x}_j)) &= \|\Phi(\boldsymbol{x}_i)-\Phi(\boldsymbol{x}_j)\|^2 \\
&= \|\Phi(\boldsymbol{x}_i)\|^2 + \|\Phi(\boldsymbol{x}_j)\|^2 - 2\Phi(\boldsymbol{x}_i)^T\Phi(\boldsymbol{x}_j) \\
&= \boldsymbol{\kappa}(\boldsymbol{x}_i,\boldsymbol{x}_i) + \boldsymbol{\kappa}(\boldsymbol{x}_j,\boldsymbol{x}_j) - 2\boldsymbol{\kappa}(\boldsymbol{x}_i,\boldsymbol{x}_j),
\end{aligned} \tag{3}$$

and the kernel version of modified Pearson correlation is given by [16]

$$\begin{aligned}
\mathcal{S}_P^{\kappa}(\Phi(\boldsymbol{x}_i),\Phi(\boldsymbol{x}_j)) &= \frac{\Phi(\boldsymbol{x}_i)^T\Phi(\boldsymbol{x}_j)}{\sqrt{\Phi(\boldsymbol{x}_i)^T\Phi(\boldsymbol{x}_i)}\sqrt{\Phi(\boldsymbol{x}_j)^T\Phi(\boldsymbol{x}_j)}} \\
&= \frac{\boldsymbol{\kappa}(\boldsymbol{x}_i,\boldsymbol{x}_j)}{\sqrt{\boldsymbol{\kappa}(\boldsymbol{x}_i,\boldsymbol{x}_i)}\sqrt{\boldsymbol{\kappa}(\boldsymbol{x}_j,\boldsymbol{x}_j)}}.
\end{aligned} \tag{4}$$

Kernel k-means (or kernel FCM) is kernel counterpart of the k-means (or FCM) [17], whose core part is to calculate the distance between the objects and the centroids of clusters in the feature space

$$\begin{aligned}
\mathcal{D}_E^{\kappa}(\Phi(\boldsymbol{x}_i),\boldsymbol{c}_k^{\Phi}) &= \|\Phi(\boldsymbol{x}_i) - \frac{1}{N_k}\sum_{l=1}^{N} u_{k,l}^m\Phi(\boldsymbol{x}_l)\|^2 \\
&= \boldsymbol{\kappa}(\boldsymbol{x}_i,\boldsymbol{x}_i) - \frac{1}{N_k}\sum_{l=1}^{N} u_{k,l}^m\boldsymbol{\kappa}(\boldsymbol{x}_i,\boldsymbol{x}_l) \\
&\quad + \frac{1}{N_k^2}\sum_{l=1}^{N}\sum_{n=1}^{N} u_{k,l}^m u_{k,n}^m\boldsymbol{\kappa}(\boldsymbol{x}_l,\boldsymbol{x}_n),
\end{aligned} \tag{5}$$

where $N_k = \sum_{l=1}^{N} u_{kl}^m$. For crisp kernel k-means, the elements in partition matrix are either zero or one and $m = 1$; for the kernel FCM, the partition matrix is the one described in Sec. 2.1 and $m \in [1,\infty)$. The procedure of kernel k-means and kernel FCM is summarized in Table 2. Other kernel-based algorithms including kernel hierarchical and kernel principle component analysis can be found in [16, 18].

2.3. Self Organizing Clustering

The Kohonen self-organizing map (SOM) is one of the most popular unsupervised clustering algorithms [19, 20] and has been reviewed in many references [2, 3]. There is another algorithm, called self-organizing oscillator networks (SOON) [21], which also belongs to self-organizing clustering family. The so-called self-organizing means that all the prototypes are attracted to the input patterns in an adaptive fashion. Here, we focus on the SOON algorithm, which

Table 2. The procedure of kernel k-means and kernel FCM

STEP 1:
Initialize a K-partition in the feature space;
STEP 2:
Calculate $\mathcal{D}_E^{\kappa}(\Phi(\boldsymbol{x}_n), \boldsymbol{c}_k^{\Phi})$ for $n = 1, ..., N$ and $k = 1, ..., K$.
STEP 3:
Update the membership matrix $\boldsymbol{U}(t)$ by

$u_{k,n} = \begin{cases} 1 & \mathcal{D}_E^{\kappa}(\Phi(\boldsymbol{x}_n), \boldsymbol{c}_k^{\Phi}) < \mathcal{D}_E^{\kappa}(\Phi(\boldsymbol{x}_n), \boldsymbol{c}_{k'}^{\Phi}) \\ 0 & otherwise \end{cases}$, for kernel k-means;

$u_{k,n} = 1/\left[\sum_{k'=1}^{K} \left(\frac{\mathcal{D}_E^{\kappa}(\Phi(\boldsymbol{x}_i), \boldsymbol{c}_{k'}^{\Phi})}{\mathcal{D}_E^{\kappa}(\Phi(\boldsymbol{x}_i), \boldsymbol{c}_k^{\Phi})}\right)^{2/(1-m)}\right]$, for kernel FCM.

STEP 4:
Repeat **STEP 2 and 3** until $\sum_{k=1}^{K} \mathcal{D}_E^{\kappa}(\boldsymbol{c}_k^{\Phi}(t), \boldsymbol{c}_k^{\Phi}(t-1)) < \epsilon$, where ϵ is a small positive number.

makes use of a biological fact that fireflies flash together exhibiting a synchronized firing in groups that physically close to each other. The basic unit of clustering in SOON is an integrate and fire (IF) oscillator representing each object in the dataset.

Suppose that $\boldsymbol{\mathcal{O}} = \{\mathcal{O}_1, ..., \mathcal{O}_N\}$ is a set of N oscillators, where each oscillator $\mathcal{O}_i$ is characterized by a phase ϕ_i and a state variable s_i, given by

$$s_i = f_i(\phi_i), i = 1, ..., K, \tag{6}$$

where each function $f_i : [0, 1] \mapsto [0, 1]$ is smooth. In [21], the function $f(\phi)$ was

$$f(\phi) = \frac{1}{b} \ln \left[1 + \left(e^b - 1\right) \phi\right].$$

Whenever s_i reaches a threshold at $s_i = 1$, the i-th oscillator fires and instantaneously reset to zero, following which the cycle repeats. The firing of all other oscillators $\mathcal{O}_j$ $(j \neq i)$ can be affected by i-th oscillator by

$$s_j(t^+) = B(s_j(t) + \epsilon_i(\phi_j)), \tag{7}$$

where $B(\cdot)$ is a limiting function to guarantee that $s_j(t)$ is confined to $[0, 1]$, mathematically expressed by

$$B(s) = \begin{cases} s & \text{if } 0 \leq s \leq 1; \\ 0 & \text{if } s < 0; \\ 1 & \text{if } s > 1. \end{cases} \tag{8}$$

The coupling strength of the i-th oscillator at a given phase ϕ_j, $\epsilon_i(\phi_j)$, is the most important concept of the SOON algorithm, mathematically expressed by

$$\epsilon_i(\phi_j) = \begin{cases} C_E \left[1 - \left(\frac{\mathcal{D}(\mathcal{O}_i, \mathcal{O}_j)}{\delta_0}\right)^2\right], & \text{if } \mathcal{D}(\mathcal{O}_i, \mathcal{O}_j) \leq \delta_0; \\ -C_I \left[\left(\frac{\mathcal{D}(\mathcal{O}_i, \mathcal{O}_j) - \delta_0}{\delta_1 - \delta_0}\right)^2\right], & \text{if } \delta_0 < \mathcal{D}(\mathcal{O}_i, \mathcal{O}_j) \leq \delta_1; \\ -C_I, & \text{otherwise,} \end{cases} \tag{9}$$

where δ_0 and δ_1 are limit distances and δ_1 is set to be five times δ_0. C_E and C_I are the maximum excitatory coupling and the maximum inhibitory coupling, respectively.

To avoid the need to compute and store the pairwise distances between any pair of objects, prototypes $\boldsymbol{\beta} = \{\beta_1, ..., \beta_K\}$ are used to represent clusters of the object in SOON-2, which is summarized in Table 3.

Table 3. The procedure of SOON-2

STEP 1:
Initialize a phases ϕ_i randomly for $i = 1, ..., N$;
Set K, and initialize the Prototypes β_k randomly for $k = 1, ..., K$;
STEP 2:
Identify the next oscillator to fire, $\{\mathcal{O}_i : \phi_i = \max_j \phi_j\}$;
Identify the close prototype to the oscillator $\mathcal{O}_i \mapsto \beta_k$;
Compute $\mathcal{D}(\beta_k, \mathcal{O}_j)$ for $\forall j \in [1, N]$;
Bring ϕ_i to threshold, and adjust other phases
$\phi_j = \phi_j + (1 - \phi_i)$ for $\forall j \in [1, N]$;
STEP 3:
for all oscillators $\mathcal{O}_j$ $(j \neq i)$ **do**
Compute state variable s_j;
Compute coupling strength $\epsilon_i(\phi_j)$;
Adjust state variable s_j using (7);
Compute compute new phase using $\phi_j = f^{-1}(s_j)$;
end for
Identify synchronized oscillators and reset their phases;
Update prototype β_k;
STEP 4:
Repeat **STEP 2 and 3** until synchronized group stabilize.

2.4. Self Splitting and Merging Clustering

Self-splitting and merging clustering is an idea in which without setting the number of clusters a priori, the algorithm will converge to a partitioning which reveals the true number of clusters and provides fairly accurate clustering results. Recently, some self splitting-merging clustering algorithms have been developed for both general purpose clustering [22, 23] and gene expression data analysis [24]. A competitive learning paradigm, called one-prototype-take-one-cluster (OPTOC) [22], was proposed in the self-splitting clustering algorithm. There are two advantages of the OPTOC that, firstly, it is not sensitive to initialization, and secondly, in many cases, it is able to find natural clusters. However, its ability to find the natural clusters depends on the determination of suitable threshold, which is difficult [24]. Being aware of the shortcoming of the OPTOC, a self-splitting-merging competitive learning (SSMCL) algorithm [24] based on the OPTOC paradigm was developed for gene expression analysis. The SSMCL initially over-clusters the whole dataset using the OPTOC principle and then merge the groups based on the second order statistical characteristics. However, although the number of clusters can be initially set to any value lager than the number of natural clusters, the SSMCL still needs to set it as close to the number of natural clusters as possible, otherwise, too much computing power will be wasted due to the unnecessary over-clustering and merging. With the similar principle as the SSMCL, over-clustering and merging, a cohesion-based self-merging (CSM) algorithm, which was reported in [23] to combine the k-means and hierarchical clustering, also faces the same problem of setting the initial number of clusters.

Here, we briefly introduce the OPTOC competitive learning paradigm. For each prototype, an online learning vecoter called asymptotic property vector (APV), $\mathcal{A}_k$, is assigned to guide the learning the k-th prototype $\mathcal{P}_k$. The APV is adapted according to

$$\mathcal{A}_k = \mathcal{A}_k + \frac{1}{n_A^k} \cdot \delta_k \cdot (\boldsymbol{x}_n - \mathcal{A}_k) \Theta(\mathcal{P}_k, \mathcal{A}_k, \boldsymbol{x}_n), \tag{10}$$

where

$$\begin{aligned} n_A^k &= n_A^k + \delta_k \cdot \Theta(\mathcal{P}_k, \mathcal{A}_k, \boldsymbol{x}_n), \\ \Theta(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}) &= \begin{cases} 1 & \text{if } \mathcal{D}(\boldsymbol{a}, \boldsymbol{b}) \leq \mathcal{D}(\boldsymbol{a}, \boldsymbol{c}) \\ 0 & \text{otherwise} \end{cases}, \end{aligned} \tag{11}$$

and

$$\delta_k = \left[\frac{\mathcal{D}(\mathcal{P}_k, \mathcal{A}_k)}{\mathcal{D}(\mathcal{P}_k, \boldsymbol{x}_n) + \mathcal{D}(\mathcal{P}_k, \mathcal{A}_k)} \right]. \quad (12)$$

The learning process for $\mathcal{P}_k$ is given by

$$\mathcal{P}_k = \mathcal{P}_k + \alpha_k \cdot (\boldsymbol{x}_n - \mathcal{P}_k)\Theta(\mathcal{P}_k, \mathcal{A}_k, \boldsymbol{x}_n) \quad (13)$$

where

$$\alpha_k = \left[1 + \frac{\mathcal{D}(\mathcal{P}_k, \boldsymbol{x}_n)}{\mathcal{D}(\mathcal{P}_k, \mathcal{A}_k)} \right]^{-2}. \quad (14)$$

The above OPTOC competitive learning paradigm is an effective technique to implement the self splitting and merging clustering.

2.5. Ensemble Consensus Clustering

Robustness is one of the desired properties of clustering algorithms, However, there is no perfect method which always gives the best results for all types of datasets. In order to enhance the robustness of clustering, the idea of ensemble consensus clustering has been proposed where the partitioning results of many clustering experiments are combined [25–33]. These partitioning results may come from different clustering algorithms, or same clustering algorithm with different parameters and initializations, or same clustering algorithm to different re-sampled permutations of the target dataset.

Although cluster ensembles have been regarded as promising methods, many obstacles have been found while combining results from different experiments. Due to the fact that clustering is unsupervised, one main problem is that it is not a straightforward task to map a specific cluster from one of the clustering results to its corresponding cluster from another clustering result. Another problem is that different clustering results may give different numbers of clusters while the correct number of clusters in unknown.

Consensus function method has been employed as an essential step in cluster ensembles. For R partitions $\{\boldsymbol{U}_1, ..., \boldsymbol{U}_R\}$, the optimal consensus partition $\boldsymbol{U}^*$ is the one which is the most similar to all of them and is mathematically given by

$$\boldsymbol{U}^* = \arg\max_{\forall \boldsymbol{P}} \sum_{j=1}^{R} \Gamma(\boldsymbol{U}, \boldsymbol{U}_j) \quad (15)$$

where $\Gamma(\cdot, \cdot)$ measures the similarity between any two partitions. This optimization problem has been noted as an NP-complete problem. There are many methods for consensus function, including relabelling and voting [33], co-association matrix [34], hypergraph methods [25], weighted kernel consensus functions [31], non-negative matrix factorization [32], greedy algorithms [27], *etc*. In all of the aforementioned methods, there are at least three steps to implement the ensemble consensus clustering as follows:

Partitions generation: R different clustering experiments are carried out to generate R partitions. The results of these partitions are all presented in a consistent form known as the partition matrix.

Relabelling: The clusters in the generated partitions are relabelled such that the corresponding clusters from different partitions are aligned.

Final consensus partition matrix generation: The relabelled partition matrices are "assembled" to generate the final consensus partition matrix.

Among these three steps, *Relabelling* and *Final consensus partition matrix generation* are the most essential parts. An example of relabelling is detailed in the steps below:

(a) A dissimilarity distance matrix $\mathbf{S}_{K\times K}$ is constructed by calculating the pairwise distance between the rows (clusters) of the matrix $\boldsymbol{U}$ and the rows of the reference matrix $\boldsymbol{U}^{ref}$.
(b) The minimum value in each of the columns is found.
(c) The maximum value of these minima is identified then the rows (clusters) from $\boldsymbol{U}$ and $\boldsymbol{U}^{ref}$ which correspond to this similarity value are mapped.
(d) The row and the column which show the aforementioned value are deleted from the similarity matrix.
(e) If all of the K rows from $\boldsymbol{U}$ and $\boldsymbol{U}^{ref}$ are mapped, the algorithm terminates, otherwise it goes back to step (2) with the reduced similarity matrix.

It is suggested that an intermediate consensus partition matrix $\boldsymbol{U}^{int(k)}$ is initialized with the values of the first partition $\boldsymbol{U}^1$, and then the other partitions are relabelled and fused with this intermediate matrix one by one while considering it as the reference at each step. Mathematically, let $\hat{\boldsymbol{U}}^r$ be the relabelled partition matrix of the partition $\mathbf{U}^r$ and let $\boldsymbol{U}^{int(k)}$ be the intermediate partition matrix after the k-th stage, i.e. after relabelling and fusing the partitions $\{\boldsymbol{U}^1, \cdots, \boldsymbol{U}^k\}$. Let the function Relabel$(\boldsymbol{U}, \boldsymbol{U}^{ref})$ denote relabelling the partition matrix U by considering $\boldsymbol{U}^{ref}$ as the reference partition. Equation (3) shows how the intermediate partition matrix can be calculated by the normal approach and the recursive approach:

$$\boldsymbol{U}^{int(k)} = \frac{1}{k}\sum_{r=1}^{k} \hat{\boldsymbol{U}}^r = \frac{1}{k}\hat{\boldsymbol{U}}^k + \frac{(k-1)}{k}\boldsymbol{U}^{int(k-1)}. \quad (16)$$

An example of generating the final consensus partition matrix is achieved by following the algorithm shown in the following steps:

(1) $\boldsymbol{U}^{int(1)} = \boldsymbol{U}^1$
(2) For $k = 2$ to R
  a. $\hat{\boldsymbol{U}}^k = \text{Relabel}(\boldsymbol{U}^k, \boldsymbol{U}^{int(k-1)})$
  b. $\boldsymbol{U}^{int(k)} = \frac{1}{k}\hat{\boldsymbol{U}}^k + \frac{k-1}{k}\boldsymbol{U}^{int(k-1)}$
(3) $\boldsymbol{U}^* = \boldsymbol{U}^{int(R)}$.

3. CLUSTERING VALIDATION

Since clustering is unsupervised classification, it is more difficult to assess than a supervised approach. Thus, the task of assessing the results of clustering algorithms can be as important as the clustering algorithms themselves. There are two functional advantages of employing the clustering validations: firstly, they can validate a clustering algorithm by comparing with other algorithms; secondly, some validity indices can provide an estimate of the number of clusters, which is crucial information for the clustering analysis. In this section, we will review some existing clustering validations.

3.1. Parametric Validity index

A parametric validity index (PVI), which employs two tunable parameters α and β to to control the proportions of objects that are involved in the calculation of the intra-cluster dissimilarities and the inter-cluster dissimilarities, was proposed in [35]. For each cluster, three spaces are defined, namely the inner space, the intra outer space and the inter outer space, representing the objects inside the cluster chosen for the calculation of both the intra-cluster dissimilarities and the inter-cluster dissimilarities, the objects inside the cluster chosen for the calculation of only the intra-cluster dissimilarities, and the objects outside the cluster chosen for the calculation

of only the inter-cluster dissimilarities, respectively. Let N_k^i, Na_k^o, Ne_k^o denote the numbers of objects in the inner space, the intra outer space and the inter outer space, respectively, for the k-th cluster. The fractions, α and β, are used to control N_k^i, Na_k^o, Ne_k^o, which can be expressed as

$$N_k^i = \lceil \alpha N_k \rceil, Na_k^o = \lceil \beta N_k \rceil, Ne_k^o = \lceil \beta (N - N_k) \rceil, \quad (17)$$

where N_k is the number of the objects in the k-th cluster, N is the number of all objects in the dataset and $\lceil \cdot \rceil$ is the ceiling operator. Both α and β can be chosen from the range of (0,1]. Thus, the inner space is $\mathcal{A}_k = \{\boldsymbol{a}_k^a | a = 1, ..., N_k^i\}$, the intra outer space is $\mathcal{B}_k^a = \{\boldsymbol{b}_k^{a,b} | b = 1, ..., Na_k^o\}$, and the inter outer space is $\mathcal{C}_k^a = \{\boldsymbol{c}_k^{a,c} | c = 1, ..., Ne_k^o\}$. The PVI is obtained by

$$\text{PVI}(K, \alpha, \beta) = \sum_{k=1}^{K} \sum_{a=1}^{N_k^i} \left(\frac{\mathcal{D}e_k^a}{\mathcal{D}a_k^a} \right), \quad (18)$$

where

$$\begin{aligned} \mathcal{D}a_k^a &= \frac{\sum_{b=1}^{Na_k^o} \mathcal{D}(\boldsymbol{a}_k^a, \boldsymbol{b}_k^{a,b})}{Na_k^o} \\ \mathcal{D}e_k^a &= \frac{\sum_{c=1}^{Ne_k^o} \mathcal{D}(\boldsymbol{a}_k^a, \boldsymbol{c}_k^{a,c})}{Ne_k^o}. \end{aligned} \quad (19)$$

3.2. Other Indices

In this section, we list five other existing validity indices. The first is the V_I [36]. The validity index V_I is the ratio of the inter-cluster separation measures and the intra-cluster scatter measures, which is mathematically expressed as

$$V_I(K) = \frac{\sum_{i=1}^{K} Ie_i}{\sum_{i=1}^{K} Ia_i}, \quad (20)$$

where K is the number of clusters. The V_I employs Ia_i, the largest dissimilarity of the minimum spanning tree (MST) for cluster i, as the intra-cluster scatter and $Ie_i = \min_{j=1, j \neq i}^{K} Ie_{ij}$, where Ie_{ij} is the the largest dissimilarity of the MST for cluster i and cluster j, as the inter-cluster separation.

The second is the *DI* [37], which is written as

$$DI(K) = \min_{1 \leq i \leq K} \left\{ \min_{1 \leq j \leq K} \left\{ \frac{\delta(\mathcal{C}_i, \mathcal{C}_j)}{\max_{1 \leq k \leq K} \{\Delta(\mathcal{C}_k)\}} \right\} \right\}, \quad (21)$$

where $\delta(\mathcal{C}_i, \mathcal{C}_j) = \min \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2$ is the maximum distance between cluster i and cluster j, $\Delta(\mathcal{C}_k)$ is the largest intra-cluster separation of cluster k. The third is the *II* [38], which is written as

$$II(K) = \left(\frac{1}{K} \times \frac{E_1}{E_K} \times D_K \right)^P, \quad (22)$$

where $E_1 = \sum_j \|\boldsymbol{x}_j - \boldsymbol{c}\|_2$ where $\boldsymbol{c}$ is the centroid of the whole dataset, $E_K = \sum_{k=1}^{K} \sum_{j \in \mathcal{C}_k} \|\boldsymbol{x}_j - \boldsymbol{c}_k\|_2$, $D_K = \max_{i,j}^{K} \|\boldsymbol{c}_i - \boldsymbol{c}_j\|_2$ and power P is constant, which is 2 in our experiments. The fourth is the *GI* [39], which is expressed as

$$GI(K) = \max_{1 \leq k \leq K} \left\{ \frac{(2 \sum_{m=1}^{M} \sqrt{\lambda_{mk}})^2}{\min_{1 \leq j \leq K} \|\boldsymbol{u}_i - \boldsymbol{u}_j\|_2} \right\}, \quad (23)$$

where M is the number of dimensions, λ_{mk} are the eigenvalues of the covariance matrix of the k-th cluster. Note that the closest *GI* value to zero suggests the best number of clusters. The fifth is the CH [40] which is given by

$$CH(K) = \frac{\left[\frac{\sum_{k=1}^{K} n_k \|\boldsymbol{c}_k - \boldsymbol{u}\|^2}{K-1} \right]}{\left[\frac{\sum_{k=1}^{K} \sum_{i=1}^{n_k} \|\boldsymbol{x}_i - \boldsymbol{c}_k\|^2}{n-K} \right]}, \quad (24)$$

where n_k is the number of memberships in the cluster k and n is the total number of the objects. The performance of above five clustering validations has been compared for gene expression dataset in [41]

4. DATASETS

There have been many benchmark gene expression datasets in the literature. People can either test their new clustering algorithms using these datasets or employ their clustering algorithms to investigate these datasets and provide new points of view in the biological context.

(a) The synthetic data set models gene expression data with cyclic behavior. Classes are modelled as genes that have peak times over the times course as presented in [42, 43].

(b) The leukemia dataset [44] consists of 38 bone marrow samples obtained from acute leukemia patients at time of diagnosis. The samples include 11 acute myeloid leukemia (AML) samples, 8 T-lineage acute lymphoblastic leukemia (ALL) samples and 19 B-lineage ALL samples. There are 999 genes in the dataset.

(c) The yeast cell cycle dataset was published by Cho *et al.* [45]. It consists of more than 6000 genes over 17 time points taken at 10 minutes intervals, where 383 genes were identified and these demonstrated consistent periodic changes in transcript level. It was commonly believed that the time course was divided into early G1, late G1, S, G2, and M phases, those 383 genes would peak at one of the five phases. In [42], a subset with 384 genes was investigated.

(d) The lymphoma dataset have three most prevalent adult lymphoid malignancies [46].

(e) The liver cancer dataset is available in [47].

5. CONCLUSIONS AND FUTURE WORK

In this paper, we systematically reviewed five clustering families representing five different clustering concepts. We also reviewed some clustering validations and collected a list of benchmark gene expression datasets. We are implementing algorithms in all of the aforementioned clustering families and investigating their performance in all of listed benchmark gene expression datasets. The performance comparison and validation will be presented in a future publication and the forthcoming conference.

6. REFERENCES

[1] D. M. Dziuda, *Data Mining for Genomics and Proteomics: Analysis of Gene and Protein Expression Data*, Wiley, 2010.

[2] R. Xu and D. II Wunsch, "Survey of clustering algorithms," *IEEE Trans. Neural Networks*, vol. 16, no. 3, pp. 645 – 678, 2005.

[3] D. X. Jiang, C. Tang, and A. D. Zhang, "Cluster analysis for gene expression data: A survey," *IEEE Trans. Know. and Data Eng.*, vol. 16, no. 11, pp. 1370–1386, 2004.

[4] Bezdek, *Pattern recognition with fuzzy objective function algorithms*, New York: Plenum, 1981.

[5] James C. Bezdek, Robert Ehrlich, and William Full, "FCM: The fuzzy c-means clustering algorithm," *Comput. Geosoci*, vol. 10, no. 2-3, pp. 191 – 203, 1984.

[6] M.E. Futschik and N.K. Kasabov, "Fuzzy clustering of gene expression data," in *Proc. IEEE Int. Conf. Fuzzy Systems, FUZZ-IEEE'02*, 2002, vol. 1, pp. 414 –419.

[7] Doulaye Dembl and Philippe Kastner, "Fuzzy c-means method for clustering microarray data," *Bioinformat.*, vol. 19, no. 8, pp. 973–980, 2003.

[8] J. W. Luo, T. Yang, and Y. Wang, "Missing value estimation for microarray data based on fuzzy c-means clustering," in *Proc. Eighth Int. Conf. High-Perf. Computing in Asia-Pacific Region*, 2005.

[9] Sanghamitra Bandyopadhyay, Anirban Mukhopadhyay, and Ujjwal Maulik, "An improved algorithm for clustering gene expression data," *Bioinformat.*, vol. 23, no. 21, pp. 2859–2865, 2007.

[10] Luis Tari, Chitta Baral, and Seungchan Kim, "Fuzzy c-means clustering with prior biological knowledge," *J. Biomed. Informat.*, vol. 42, no. 1, pp. 74 – 81, 2009.

[11] R.D. Pascual-Marqui, A.D. Pascual-Montano, K. Kochi, and J.M. Carazo, "Smoothly distributed fuzzy c-means: a new self-organizing map," *Pattern Recognition*, vol. 34, no. 12, pp. 2395 – 2402, 2001.

[12] Shuta Tomida, Taizo Hanai, Hiroyuki Honda, and Takeshi Kobayashi, "Analysis of expression profile using fuzzy adaptive resonance theory," *Bioinformat.*, vol. 18, no. 8, pp. 1073–1083, 2002.

[13] Y. Mao, X. B. Zhou, D. Y. Pi, Y. X. Sun, and Stephen T. C. Wong, "Multiclass cancer classification by using fuzzy support vector machine and binary decision tree with gene selection," *J. Biomed. Biotech.*, vol. 2005, no. 2, pp. 160–171, 2005.

[14] K.-R. Muller, S. Mika, G. Ratsch, K. Tsuda, and B. Scholkopf, "An introduction to kernel-based learning algorithms," *IEEE Trans. Neural Networks,* , vol. 12, no. 2, pp. 181 –201, 2001.

[15] Michael P. S. Brown, William Noble Grundy, David Lin, Nello Cristianini, Charles Walsh Sugnet, Terrence S. Furey, Manuel Ares, and David Haussler, "Knowledge-based analysis of microarray gene expression data by using support vector machines," *Proc. Nat. Academy Sci.*, vol. 97, no. 1, pp. 262–267, 2000.

[16] Jie Qin, Darrin P. Lewis, and William Stafford Noble, "Kernel hierarchical gene clustering from microarray expression data," *Bioinformat.*, vol. 19, no. 16, pp. 2097–2104, 2003.

[17] Inderjit S. Dhillon, Yuqiang Guan, and Brian Kulis, "Kernel k-means: spectral clustering and normalized cuts," in *Proc. Tenth ACM SIGKDD Int. Conf. Know. disc. data mining*, New York, NY, USA, 2004, KDD '04, pp. 551–556, ACM.

[18] Z. G. Liu, D. C. Chen, and H. Bensmail, "Gene expression data classification with kernel principal component analysis," *J. Biomed. Biotech.*, vol. 2005, no. 2, pp. 155–159, 2005.

[19] T. E. Kohonen, *Self-Organizing Maps,*, New York: Springer-Verlag, 1997.

[20] P. Tamayo, D. Slonim, J. Mesirov, Q. Zhu, S. Kitareewan, E. Dmitrovsky, E. S. Lander, and T. R. Golub, "Interpreting patterns of gene expression with self-organizing maps: Methods and application to hematopoietic differentiation," *Proc. Nat. Academy Sci. USA*, vol. 96, no. 6, pp. 2907–2912, 1999.

[21] Sameh A. Salem, Lindsay B. Jack, and Asoke K. Nandi, "Investigation of self-organizing oscillator networks for use in clustering microarray data," *IEEE Trans. NanoBio.*, vol. 7, no. 1, pp. 65–79, 2008.

[22] Ya-Jun Zhang and Zhi-Qiang Liu, "Self-splitting competitive learning: a new on-line clustering paradigm," *IEEE Trans. Neural Networks*, vol. 13, no. 2, pp. 369 –380, mar 2002.

[23] Cheng-Ru Lin and Ming-Syan Chen, "Combining partitional and hierarchical algorithms for robust and efficient data clustering with cohesion self-merging," *IEEE Trans. Know. Data Eng.*, vol. 17, no. 2, pp. 145 – 159, 2005.

[24] Shuanhu Wu, A.W.-C. Liew, Hong Yan, and Mengsu Yang, "Cluster analysis of gene expression data based on self-splitting and merging competitive learning," *IEEE Trans. Inf. Tech. Biomed.*, vol. 8, no. 1, pp. 5 –15, 2004.

[25] A. Strehl and J. Ghosh, "Cluster ensembles C a knowledge reuse framework for combining multiple partitions',," *J. Machine Learning Research,*, vol. 3, pp. 583–617, 2002.

[26] Stefano Monti, Pablo Tamayo, Jill Mesirov, and Todd Golub, "Consensus clustering: A resampling-based method for class discovery and visualization of gene expression microarray data," *Machine Learning*, vol. 52, pp. 91–118, 2003.

[27] V. Filkov and S. Skiena, "Integrating microarray data by consensus clustering," in *Proc. Int. Conf. Tools with Art. Intell.*, pp. 418–426, 2003.

[28] Xiaohua Hu and Illhoi Yoo, "Cluster ensemble and its applications in gene expression analysis," in *Proc. Second Conf. Asia-Pacific bioinformat.*, APBC'04,- v. 29, pp. 297–302, Australian Computer Society.

[29] Stephen Swift, Allan Tucker, Veronica Vinciotti, Nigel Martin, Christine Orengo, Xiaohui Liu, and Paul Kellam, "Consensus clustering and functional interpretation of gene-expression data," *Genome Biology*, vol. 5, no. 11, pp. R94, 2004.

[30] R. Avogadri and G. Valentini, "Ensemble clustering with a fuzzy approach," *Studies in Comput. Intell. Superv. and Unsuperv. Ensemble Methods their Appl.*, v. 126, pp 49-69, 2008.

[31] S. Vega-Pons, J. Correa-Morris, and J. Ruiz-Shulcloper, "Weighted cluster ensemble using a kernel consensus function," *Lecture Notes in Computer Science, Heidelberg: Springer.*, vol. 5197, pp. 195–202, 2008.

[32] W. Wang, "An improved non-negative matrix factorization algorithm for combining multiple clusterings," in *2010 Int. Conf. Machine Vision Human-machine Interface*, pp. 604–607, 2010.

[33] H. G. Ayad and M. S. Kamel, "On voting-based consensus of cluster ensembles," *Pattern Recognition*, vol. 43, pp. 1943-C1953, 2010.

[34] A. Fred and A. K. Jain, "Data clustering using evidence accumulation',," in *Proc. Sixteenth Int. Conf. Pattern Recognition (ICPR)*, pp. 276–280, 2002.

[35] R. Fa and A. K. Nandi, "Parametric validity index of clustering for microarray gene expression data," in *IEEE Int. Workshop Machine Learning for Sig. Process. 2011 (MLSP 2011)*, 2011.

[36] Sameh A. Salem and Asoke K. Nandi, "Development of assessment criteria for clustering algorithms," *Pattern Anal. and Appl.*, vol. 12, no. 1, pp. 79–98, 2009.

[37] J. C. Dunn, "A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters," *J. Cyber.*, vol. 3, no. 3, pp. 32–57, 1973.

[38] U Maulik and S Bandyopadhyay, "Performance evaluation of some clustering algorithms and validity indices," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 12, pp. 1650–1654, 2002.

[39] Benson S. Y. Lam and Hong Yan, "Assessment of microarray data clustering results based on a new geometrical index for cluster validity," *Soft Computing*, vol. 11, no. 4, pp. 341–348, 2007.

[40] T. Calinski and J. Harabasz, "A dendrite method for cluster analysis," *Communications in Statistics - Theory and Methods*, vol. 3, no. 1, pp. 1–27, 1974.

[41] R. Fa and A. K. Nandi, "Comparisons of validation criteria for clustering algorithms in microarray gene expression data analysis," in *Proc. Second Int. Workshop Genomic Signal Process. (GSP2011)*, 2011.

[42] K. Y. Yeung, C. Fraley, A. Murua, A. E. Raftery, and W. L. Ruzzo, "Model-based clustering and data transformations for gene expression data," *Bioinformat.*, vol. 17, no. 10, pp. 977–987, 2001.

[43] L. P. Zhao, R. Presntice, and L. Breeden, "Statistical modelling of large microarray data sets to identify stimulus-response profiles," *Proc. Natl. Acad. Sci. (PNAS)*, vol. 98, no. 10, pp. 5631–5636, 2001.

[44] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, and E. S. Lander, "Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring," *Science*, vol. 286, no. 5439, pp. 531–537, 1999.

[45] R. J. Cho, M. J. Campbell, E. A. Winzeler, L. Steinmetz, A. Conway, L. Wodicka, T. G. Wolfsberg, A. E. Gabrielian, D. Landsman, D. J. Lockhart, and R. W. Davis, "A genome-wide transcriptional analysis of the mitotic cell cycle," *Mol. Cell*, vol. 2, no. 1, pp. 65–73, 1998.

[46] Alizadeh et al., "Distinct types of diffuse large b-cell lymphoma identified by gene expression profiling," 2000, Nature, 403(6769):pp. 503–511. http://llmpp.nih.gov/lymphoma/index.shtml.

[47] Chen X et al., "Gene expression patterns in human liver cancers," 2002, Mol. Biol. Cell 13(6): pp. 1929–1939 , http://www.ncbi.nlm.nih.gov/, GSE3500 record.