Message Passing Clustering with Stochastic Merging Based on Kernel Functions

Huimin Geng
University of Nebraska, huimingeng@unmc.edu

Xutao Deng
University of Nebraska at Omaha, xdeng@mail.unomaha.edu

Hesham H. Ali
University of Nebraska at Omaha, hali@mail.unomaha.edu

Message Passing Clustering with Stochastic Merging Based on Kernel Functions

Huimin Geng
Department of Pathology and Microbiology
University of Nebraska Medical Center
huimingeng@unmc.edu

Xutao Deng
College of Information Science and Technology
University of Nebraska at Omaha
xdeng@mail.unomaha.edu

Hesham H. Ali
Department of Computer Science
University of Nebraska at Omaha
hali@mail.unomaha.edu

ABSTRACT

In this paper, we propose a new Stochastic Message Passing Clustering (SMPC) algorithm for clustering biological data based on the Message Passing Clustering (MPC) algorithm, which we introduced in earlier work. MPC has shown its advantage when applied to describing parallel and spontaneous biological processes. SMPC, as a generalized version of MPC, extends the clustering algorithm from a deterministic process to a stochastic process, adding three major advantages. First, in deciding the merging cluster pair, the influences of all clusters are quantified by probabilities, estimated by kernel functions based on their relative distances. Second, the proposed algorithm property resolve the “tie” problem, which often occurs for integer distances as in the case of protein interaction data. Third, clustering can be undone to improve the clustering performance when the algorithm detects objects which don’t have good probabilities inside the cluster and moves them outside. The test results on colon cancer gene-expression data show that SMPC performs better than the deterministic MPC.

Supplementary information: http://bioinformatics.ist.unomaha.edu/~hgeng/.

Keywords

Clustering, message passing clustering, hierarchical clustering, stochastic processes, kernel functions.

INTRODUCTION

Ten years ago with traditional methods, molecular biologists had to work on a “one gene in one experiment” basis. It was difficult to achieve a high throughput of gene expression and to see the complete picture of gene functions. Now, with the rapidly developing microarray technology, scientists are able to determine the expression levels of thousands of genes in a single experiment (Holter *et al.*, 2000; Eisen *et al.*, 1998; Phimister *et al.*, 1999). However, the huge data sets generated from microarray experiments present us the great challenge of interpreting and analyzing the data. It has become increasingly clear that simply generating the data is not enough; one must be able to extract meaningful information about the system being studied. This requires more efficient and more accurate data analysis approaches and encourages more researchers to be devoted to this field.

The term *clustering* refers to a wide variety of unsupervised methods for organizing multivariate data into distinct groups with similar patterns. Clustering is a fundamental technique which has numerous applications in expression-data analysis, where clues to unknown gene functions may be inferred from clusters of genes similarly expressed across many samples (Eisen *et al.*, 1998), and elsewhere in biology, e.g. phylogenetic tree building and regulatory network reconstruction (Durbin *et al.*, 1998).

Among the best-known and most powerful methods of clustering are those involving Hierarchical Clustering (HC) (reviewed in Everitt *et al.*, 2001). HC is conceptually simple but often outperforms algorithms that employ complex data structures and data flows such as K-means, Self Organizing Maps (SOM), and Finite Mixtures (Ball *et al.*, 1967; Lance *et al.*, 1967; Herwig *et al.*, 1999; Kohonen, 1997; Tamayo *et al.*, 1999; Everitt *et al.*, 2001). HC arranges the data into a tree structure which can be easily viewed and understood, and the hierarchical structure provides potentially useful information about the relationships between clusters. It is one of the most widely accepted by biologists. However, HC has some inherent problems dealing with the data generated from biological processes (Alon *et al.*, 1999; Arnau *et al.*, 2005; Geng *et al.*, 2004). HC always puts a priority on global distance; that is, in each step only the two clusters which have the globally lowest distance (or highest similarity) are merged and the clusters are generated sequentially. This mechanism of HC may produce counterintuitive clusters and clusters with more singletons, and may not be suitable for biological data sets because biological processes are often parallel by nature (Geng *et al.*, 2005).

To overcome these problems and to capture biological features, Geng *et al.* proposed a new clustering algorithm, Message Passing Clustering (MPC), which simulates spontaneous and parallel biological processes (Geng *et al.*, 2004; Geng *et al.*, 2005). Inspired by real-life situations in which people in large gatherings can form groups by exchanging information, MPC allows data objects to communicate with each other, generates clusters in parallel so that the global distances and local distances are well-balanced, and hence improves the performance of clustering. The method of MPC has been successfully applied to clustering gene-expression data and to building phylogenetic trees (Geng *et al.*, 2005). The resulting dendrograms have shown its superiority over other techniques such as HC, neighbor joining (Felsenstein, 1989; Saitou and Nei, 1987), K-means, CLICK (Sharan and Shamir, 2000), and SOM.

But there are still some features that the deterministic algorithms cannot offer due to their deterministic nature:

1). *Breaking ties*: Ties often occur when the distances are integers as in the case of protein interaction data (MacCuish *et al.*, 2001; Takezaki, 1998; Arnau *et al.*, 2005). The ties in HC can be a serious problem because the final clustering solution depends on the index of the nodes. MPC can be better for dealing with ties. Considering a situation where we have four nodes *a, b, c* and *d*, the distance between *a* and *b* and the distance between *c* and *d* are equally minimum, and the basic MPC algorithm merges (*a, b*) and (*c, d*) in parallel. However, MPC can't figure out the situation when the ties happen on the same cluster, i.e., if both cluster *b* and cluster *c* are nearest to cluster *a*, then cluster *a* will pick the one which has the lower index for merging. A better mechanism would be to choose the merging neighbor by coin tosses because this is independent of the arbitrary index of nodes. In this paper, we generalize this idea in which all neighbors are allocated a probability to merge based on their distances to a given node.

2). *Ensemble influence*: By ensemble we mean the set of all nodes (clusters). In HC and MPC, finding the minimum distance is the greedy approach that ignores the influence of all other clusters on merging two clusters. We argue that the ensemble influence should be considered when deciding merging cluster pairs. Besides the first nearest neighbor, the second, the third, etc., may also be good candidates if there are no significant differences in terms of distances between them and the target cluster. This phenomenon is often overlooked by traditional HC or the basic MPC clustering techniques. It is more reasonable to distribute the merging probability over the ensemble. We use kernel functions generated by each cluster to estimate the ensemble probability distribution, and we can use a random switch to determine which cluster is to be selected based on the probabilities.

3). *Undoing clusters*: A serious problem with hierarchical methods, either divisive or agglomerative, is that their operations, once made, are irrevocable (Haukins *et al.*, 1982; Kaufman *et al.*, 1990). Rooted in the greedy feature of hierarchy methods, this "irrevocable" problem may lead to poor clustering results. The probability estimation technique (similar to the stochastic merging mechanism) can be applied to removing outliers if their probabilities staying inside are no longer satisfactory, i.e. below a certain threshold.

To include those features, we propose a generalized version of MPC, MPC with stochastic merging (SMPC), which extends the clustering algorithm from a deterministic process to a stochastic process. The proposed technique can be applied to any agglomerative clustering algorithm such as HC, MPC and SPC (superparamagnetic clustering) (Getz *et al.*, 2000).

This article is organized as follows. The next section gives a brief description of the deterministic MPC method, including the basic algorithm, the properties, the performance, and potential extensions. In section 3, we introduce kernel functions and derive the merging probabilities based on kernel functions. Section 4 presents one artificial data to demonstrate the results of SMPC and one real-data case study with colon cancer microarray data. Section 5 summarizes the SMPC method and section 6 provides a discussion.

MESSAGE PASSING CLUSTERING

We introduce MPC first as the background for SMPC. MPC is inspired by real-life situations. Suppose we have a party where people don't know one another in the beginning. Then each person may look around and talk to another person to see if they share some common interest. If so, they will continue the conversation and other people with the same interest may join this group later. It is often the case that after a while a set of talking groups are formed at the party. This communication model shows a parallel and spontaneous clustering process by exchanging information between people. We can see that many biological examples share great similarity with this party model. In the MPC algorithm, we employ the concept of message passing to represent the information exchanging processes between data objects.

Basic Algorithm of MPC

Algorithm 2.1: MPC main function

```
Inputs: M-dimensional vectors $X_1$,$X_2$,…,$X_n$
Outputs: A partition of $X_1$,$X_2$,…,$X_n$ into K clusters

MPC ($X_1$, $X_2$,…,$X_n$)
    Initialize ($X_1$, $X_2$,… $X_n$);
    while (Num_Clusters>K)
        for (each cluster $C_i$) //message sending
            $C_j$=Find_Nearest_Neighbor ($C_i$);
            Msg_Send ($C_i$,$C_j$);
    for (each cluster Ci)//message receiving
        $C_j$=Msg_Rcv ($C_i$) // check message
        if ($C_j$ not equal to NULL){
            New Cluster C'= Merge ($C_i$, $C_j$);
            Num_Clusters= Num_Clusters-1;
```

Algorithm 2.1: Subroutines

```
Initilize ($X_1$, X2,…,$X_n$)
    Num_Clusters=n; //Number of Clusters
    for (each cluster $C_i$)
        $C_i$=$X_i$; // Assign the vector to a cluster
        $C_i$.FROM=NULL;
        $C_i$.TO=NULL;

Msg_Send(Ci, Cj) //send a message from $C_i$ to $C_j$
    $C_i$.TO=j;
    $C_j$.FROM=i;

Msg_Rcv (Ci) //check the message box
    if ( $C_i$.FROM=$C_i$.TO=j) //find mutually
        return $C_j$;
    else
        return NULL;
```

Like all clustering algorithms, the input to the MPC program is a set of *M*-dimensional vectors with size *N* and the output is a partition of the *N* vectors into *K* clusters where *K* can be any number from 1 to *N*. The key idea of MPC is to allow vectors to communicate with each other so that the clusters can be formed in parallel.

Initially each vector is a cluster by itself, so the number of clusters is *N*. During the clustering process, each cluster will send a message to its nearest neighbor (the cluster which has the maximum similarity to the sending cluster) by calling function ***Msg_Send()***. Each cluster C_i is associated with two special memory cells, ***C_i.TO*** and ***C_i.FROM***. These two cells function as a message box with C_i.*TO* storing the outgoing address and C_j.*FROM* storing the incoming address. After sending messages, each cluster checks the message box by calling ***Msg_Rcv()***. If the outgoing and incoming addresses are the same, a mutually nearest neighbor (C_i is the nearest neighbor of C_j and vice versa) is found such that the two clusters merge into one. This process is repeated until the specified number of clusters *K* is reached. The program of MPC and the supplementary data can be obtained at http://bioinformatics.ist.unomaha.edu/~hgeng/.

Properties of MPC

While it appears that MPC and HC may produce similar outputs, we have proved that, more often than not, the output clusters of the two approaches differ in favor of the MPC method. MPC has the same advantage as the HC method: the algorithm is easy to understand and to implement; and the resulting clusters have a tree structure which is easily viewed and broken at any point and provides useful information about the relationship among clusters. Moreover, MPC is superior to HC because it takes into account both local and global data structure and takes advantage of communication among data objects so that clusters can be generated in parallel. MPC has the following properties:

Property 1: *The number of new clusters produced at each round of message exchanging is from 1 to n/2, where n is the number of clusters at the previous step.*

Property 2: *Comparing MPC and HC in various similarity criteria, the following propositions hold: 1) In the case of centroid linkage, the clustering dendrograms from MPC and HC are generally different. 2) In the case of single, complete, or average linkage, the final clustering dendrograms from MPC and HC are equivalent, but intermediate clusters from MPC and HC are generally different. If the number of final clusters is specified in advance, MPC and HC may yield different solutions.*

Property 1 shows clusters merge in an ideally exponential fashion so that MPC is conceptually fast, reflecting its parallel process. Property 2 concludes how the results of MPC and HC differ based on different similarity criteria. Refer to Geng *et al.* (2004) for the proofs of the properties.

Performance of MPC

We predict MPC should outperform most clustering algorithms when applied to biological data sets. The results from the case studies in building phylogenetic trees and in clustering gene expression data confirmed our hypotheses. The real data representing the relatedness of DNA sequences of 34 strains of nine species of *Mycobacterium* was used as the input to the MPC program, and in the output, biologically relevant topology for different *Mycobacterium* species was reflected in the phylogenetic tree. In another case study, the gene expression data from the Stanford yeast cell cycle database was used to compare the solution generated from the MPC method to solutions from other clustering methods such as *K* means, SOM and CLICK. MPC beats all other algorithms by providing a partition nearest to the “true” solution.

Extension of MPC

One advantage of MPC is its flexible structure which allows future development under the same logic frame. For example, if the features (attributes) consist of a mixture of data types such as nominal, ordinal, and scalar data, the distance between two data objects is not well-defined and this problem is algorithm-independent. But MPC can handle it by making a small modification: sending multiple messages according to the different data types of the attributes for each data object. This way, the messages are handled independently and clusters are formed based on all messages the objects sent and received. Another promising application of MPC is that by assigning weights to attributes and attaching information of attributes as part of the message, we can cluster data objects according to different interests or varying interest in the clustering process. The attribute we are most interested in is the dominant attribute with the highest weight.

We have implemented an extended version of MPC with Adaptive Feature Scaling (AFS), which adaptively updated the feature weights during the clustering process so that certain features (signals) are strengthened while others (noises) are diminished. The program of MPC with AFS and the supplementary data can be obtained at http://bioinformatics.ist.unomaha.edu/~hgeng/.

MPC WITH STOCHASTIC MERGING

As described in previous section, we can extend the basic MPC to reach different requirements and focuses. Here and in the following sections, we present another extension of MPC, SMPC, which allows one cluster to merge with another based on the probabilities. The algorithm for SMPC is built on the basic MPC, but we add a stochastic process to select the pair of merging clusters. Kernel functions are used to estimate the probability densities at each point in the space, and random switch technique is used to determine which cluster is to be selected based on the probabilities.

Kernel Density Estimator

From the definition of a probability density, if the random variable X has density f, then

$$f(x) = \lim_{w \to 0} \frac{1}{2w} P(x - w < X < x + w) . \tag{1}$$

For any given w, a naïve estimator of $P(x - w < X < x + w)$ is the proportion of the observations $X_1, X_2, \ldots, X_n$ falling in the interval $(x - w, x + w)$, that is,

$$\hat{f}(x) = \frac{1}{2wn} [\#\text{ of } X_{1,} X_2, \ldots, X_n \text{ falling in } (x - w, x + w)] . \tag{2}$$

If we introduce a weight function W given by

$$W(x, X_i) = \begin{cases} \frac{1}{2w} & \text{if } |x - X_i| < w \\ 0 & \text{otherwise} \end{cases} , \tag{3}$$

Then the naïve estimator (2) can be rewritten as:

$$\hat{f}(x) = \frac{1}{n} \sum_{i=1}^{n} W(x, X_i) . \tag{4}$$

This lead naturally to the kernel estimator defined as

$$\hat{f}(x) = \frac{1}{n}\sum_{i=1}^{n} K(x, X_i), \tag{5}$$

where $\hat{f}(x)$ is a probability density estimator at point *x*, and *K* is known as the *kernel function* which is a function of *x* and satisfies the condition $\int_{-\infty}^{\infty} K(x, X_i)dx = 1$. There are many kernel functions; a simple one is *W(x, X$_i$)* given by (3), called *rectangular*. Another commonly used one is *Gaussian,* defined as:

$$K(x, X_i) = \frac{1}{w\sqrt{2\pi}} e^{-\frac{1}{2w^2}(x-X_i)^2}, \tag{6}$$

where *w* is known as the *bandwidth parameter.*

The kernel estimator in (5) tells us that an instance at X_i generates a kernel function *K(x, X$_i$)* that assigns a probability to each point *x* in the space; the density estimate as a whole is just the normalized sum of all of the little kernel functions. Figure 1 gives an example of the graphs of probability density estimates.

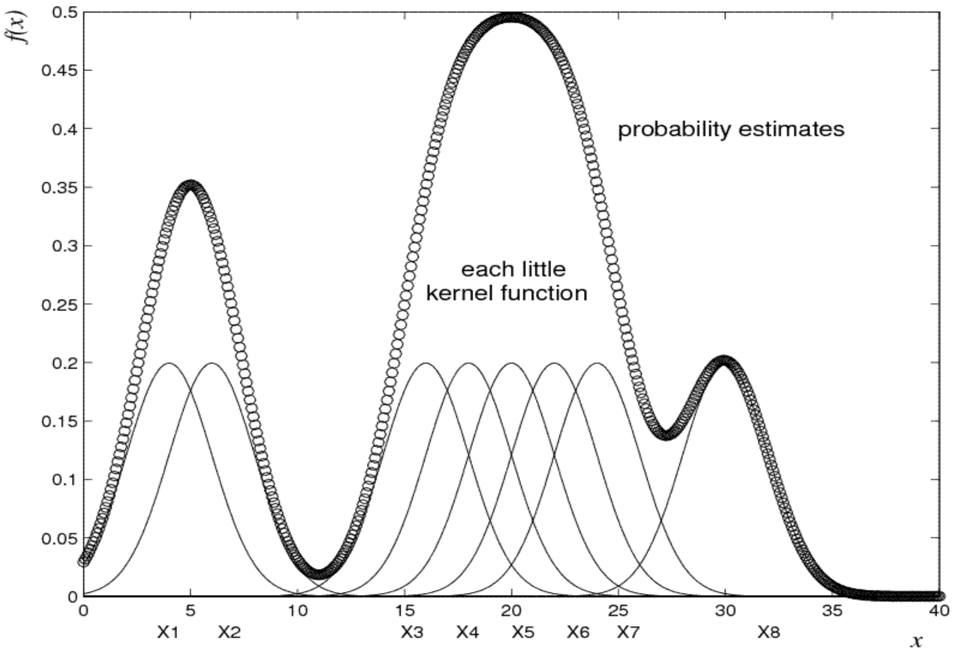

Figure 1. Probability Density Estimates Based on Little Gaussian Kernel Functions

In the example of Figure 1, there are eight instances, X_1, X_2, …, X_8, and each instance generates a Gaussian kernel function which assigns a probability density at each point *x* in the entire space. The probability estimate at point *x* is the summation of all the probabilities assigned by each kernel function. That is, $\hat{f}(x) = \frac{1}{8}\sum_{i=1}^{8}\frac{1}{w\sqrt{2\pi}} e^{-\frac{1}{2w^2}(x-X_i)^2}$.

Kernel Functions Based on the Distance

Now we employ kernel functions with the clustering problem. Given a set of *n* multidimensional data objects, suppose that at some step during the clustering process, *k* clusters {X_1, X_2, $\ldots$, X_k} remain, $1 \le k \le n$. The kernel function normally depends only on the distance *D*(*X*, X_i) from the cluster *X* to the cluster X_i, so we can rewrite the kernel functions in terms of the distance. Formula (7) and (8) are the rectangular and Gaussian kernel functions, respectively.

$$K(X, X_i) = \begin{cases} \frac{1}{2w} & \text{if } D(X, X_i) < w \\ 0 & \text{otherwise} \end{cases}, \quad X,\ X_i \in \{X_1, X_2, ..., X_k\} \text{ and } X \ne X_i \tag{7}$$

$$K(X,X_i)=\frac{1}{w\sqrt{2\pi}}e^{-\frac{1}{2w^2}D(X,X_i)^2}, \; X, X_i \in \{X_1, X_2, ..., X_k\} \text{ and } X \neq X_i \tag{8}$$

If we use the rectangular kernel function to estimate the probabilities and set the parameter *w* in each iteration of merging as the minimum distance of all the distances between *X* and *X*X_i*'s,* where $X, X_i \in \{X_1, X_2, ..., X_k\}$ and $X \neq X_i$, then the stochastic MPC is reduced to the deterministic MPC. The bandwidth parameter, *w,* affects the kernel density estimates. Figure 2 shows this effect with an example of two-dimensional data. Making *w* too small results in a very spiky estimate (Figure 2a), while making it too large results in losing the structure altogether (Figure 2d). In Figure 2 (b and c), a medium value of *w* gives a very good reconstruction. How to choose a suitable value for *w* depends on the data, the goal, and the technique. Cross-validation (Russell *et al.,* 2003) is one of the techniques for finding a good value for *w.*

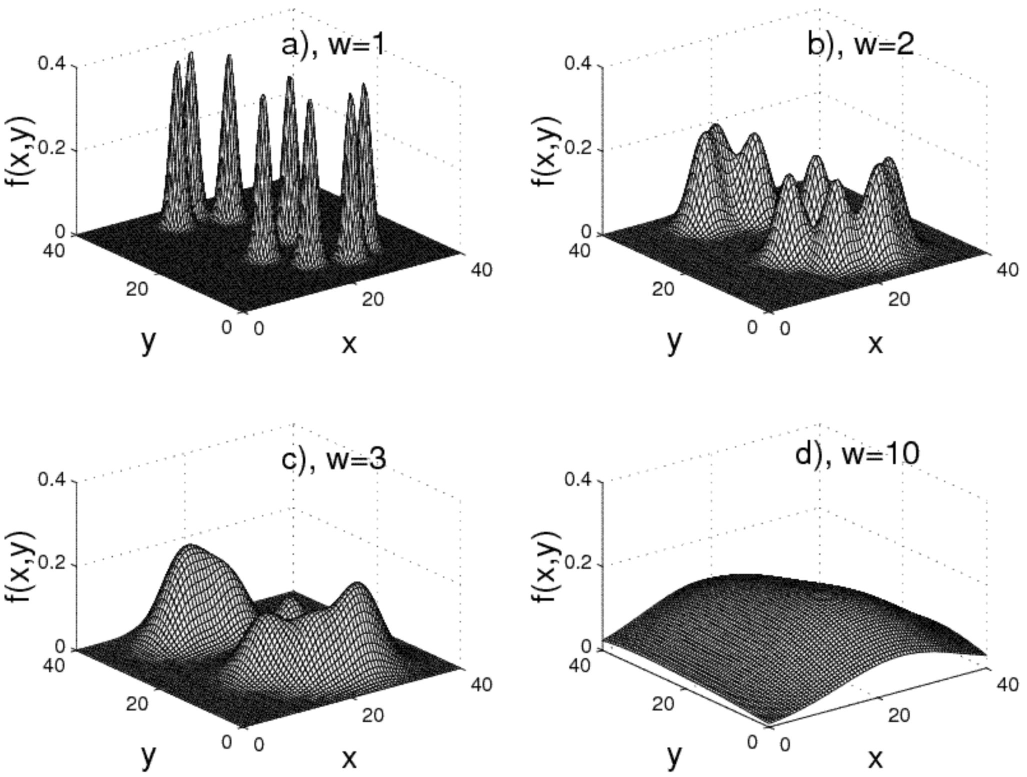

Figure 2. Demonstration of Kernel Density Estimates Using Gaussian Kernels with a) *w*=1, b) *w*=2, c) *w*=3, d) *w*=10.

Stochastic Merging

The probability estimate at point *x* is the summation of all the probabilities assigned by each kernel function at *x.* In other words, the probability that X_i's kernel function generates at point *x* is a proportion of the final probability estimate at point *x.* It's straightforward to consider X_i's merging probability with *X* (which is at point *x*) as the proportion of X_i's contribution to the total density estimate at point *x*:

$$P_{merge}(X,X_i)=\frac{K(X,X_i)}{\sum_{i=1}^{k}K(X,X_i)}=\frac{e^{-\frac{1}{2w^2}D(X,X_i)^2}}{\sum_{i=1}^{n}e^{-\frac{1}{2w^2}D(X,X_i)^2}}, \quad \text{where } X, X_i \in \{X_1, X_2, ..., X_k\} \textit{ and } X \neq X_i \tag{9}$$

Obviously, the summation of the probability that each cluster X_i is to merge with the target cluster *X* is $\sum_{i=1}^{k}P_{merge}(X,X_i)=1.$

Instead of giving one cluster entirely the power to be merged with the target, we distribute the probability over all clusters based on their relative distances to the target *X*. We still honor the one nearest to the target by assigning it the highest probability. If two clusters are both the nearest, then they are assigned equal probability. The clusters far away from the target are given a low probability of merging with the target.

Knowing the probabilities, we can use random switch technique to implement the stochastic merging (Ross, 2002). To generate a value from a discrete probability distribution

$$P_{merge}(X, Y = X_i) = p_i, \quad i = 1,\ 2,\ ...,\ n, \quad \sum_{i=1}^{n} p_i = 1, \tag{10}$$

we generate a random number *U*, which is uniformly distributed over (0, 1), and set

$$Y = \begin{cases} X_1 & \text{if } U < p_1 \\ X_2 & \text{if } p_1 \le U < p_1 + p_2 \\ \vdots & \\ X_n & \text{if } \sum_{j=1}^{n-1} p_j \le U < \sum_{j=1}^{n} p_j \end{cases} . \tag{11}$$

Since $P\{Y = X_i\} = P\{\sum_{j=1}^{i-1} p_j \le U < \sum_{j=1}^{i} p_j\} = p_i$, *X* has the desired distribution. In our case, if *U* falls in the interval $(\sum_{j=1}^{i-1} p_j, \sum_{j=1}^{i} p_j)$, then we choose the cluster X_i for merging.

RESULTS

Artificial Data

We use a set of artificial data to demonstrate the effects of stochastic processes in clustering. Suppose that there are five nodes in a plane: ***a***(0, 1), ***b***(0, 0), ***c***(0, -1), ***d***(1, 1), and ***e***(0.5, 1.87). The node ***a*** is equally nearest to nodes ***b***, ***d*** and ***e***, so ties happen in those three pairs of nodes. We apply both SMPC and MPC to this set of data. The resulting clusters are shown in Figure 3.

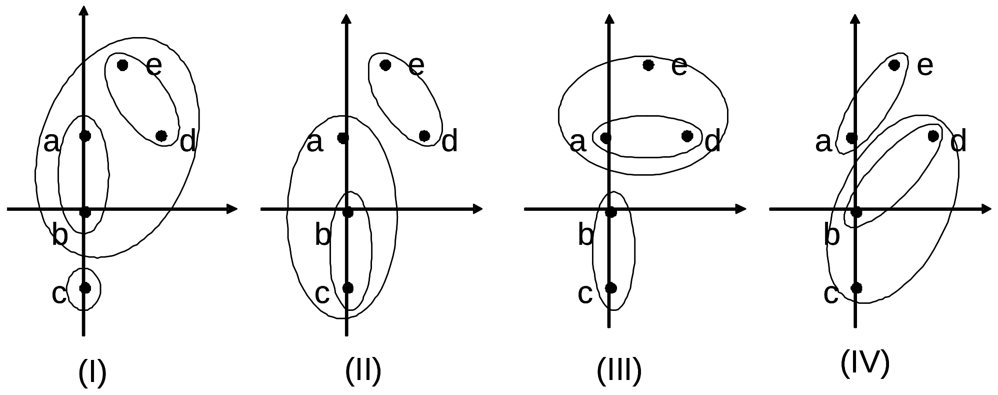

Figure 3. Clustering Results from SMPC and MPC: (I) - (IV) are four possible solutions given by SMPC, while (I) is the only solution given by MPC

With MPC, only one solution (Figure 3(I)) is given, because MPC is a deterministic method which assigns only one node (***b*** in our case) 100% probability to merge with ***a***. If we use SMPC, more possible solutions are generated. SMPC assigns equal probability to the nodes in ties, so ***a*** can not only be grouped with ***b*** as in (I), but also with ***d*** in (III) and ***e*** in (IV). Also, the parallel merging of ties happening in distinct clusters are well-illustrated in both MPC and SMPC, as in the pairs of (a, b) and (d, e) in (I), (b, c) and (d, e) in (II), and (a, e) and (b, d) in (III). Another observation is that due to characteristics of the stochastic process, the lowest indexed node ***a*** is not necessarily grouped with some node b, d or e at the first step. As shown in (II), ***b*** is attracted to ***c,*** but not to ***a***, at the first step.

Viewing this example, there is a singleton in the solution (Figure 3 (I)) generated by the deterministic MPC. Such singletons are not good in clustering (reviewed in Everitt *et al.*, 2001). SMPC gives more possible solutions (Figure 3 (II), (III), (IV)), and those solutions have no singletons. SMPC provides other points of view of how data objects are related to each other, and this is especially useful in studying the protein interactive data where the entries are integer values and ties often happen (Dress *et al.*, 2001; Arnau *et al.*, 2005).

Colon Cancer Data

We apply SMPC to colon cancer microarray data to see the potential improvement of the clustering results as compared to those from the deterministic MPC. This set of data contains 62 samples over 2000 genes (Alon *et al.*, 1999). Among 62 samples, 40 are tumors and 22 are normal samples. Another way to divide these 62 samples into two subgroups is using different protocols: 22 samples are prepared using Protocol A, and 40 samples using Protocol B (Getz *et al.*, 2000). We perform clustering on samples with genes as features using both MPC and SMPC and see which one can better separate samples in terms of normal and tumor or protocol A and protocol B.

We use the symbol "----" to represent 11 *protocol A normal* samples, "-" for 11 *protocol B normal* samples, "****" for 11 *protocol A tumors,* and "*" for 29 *protocol B tumors.* In SMPC, we set *w*=10 because it is a medium value according to the distances between two data objects (10^1 magnitude for average) so that we have a good reconstruction. Figure 4 shows the clustering results for MPC and SMPC. We break the branch at the point indicated by the red dash line, so that two clusters remain, each representing the normal group or tumor group from one aspect and the protocol A group or protocol B group from another aspect. We compare the classification rate (using cross-validation method) between the two methods in Table 1, which shows SMPC improves the classification precision from 62.9% to 85.5% if normal/tumor is the criterion. If protocol A/B is the criterion to distinguish the samples, there is no significant difference between the solutions given by SMPC (59.7%) and by MPC (58.1%). This makes sense, since no matter which protocol is used to prepare the sample, the characteristic of the sample doesn't change, and the pattern of gene expression doesn't change.

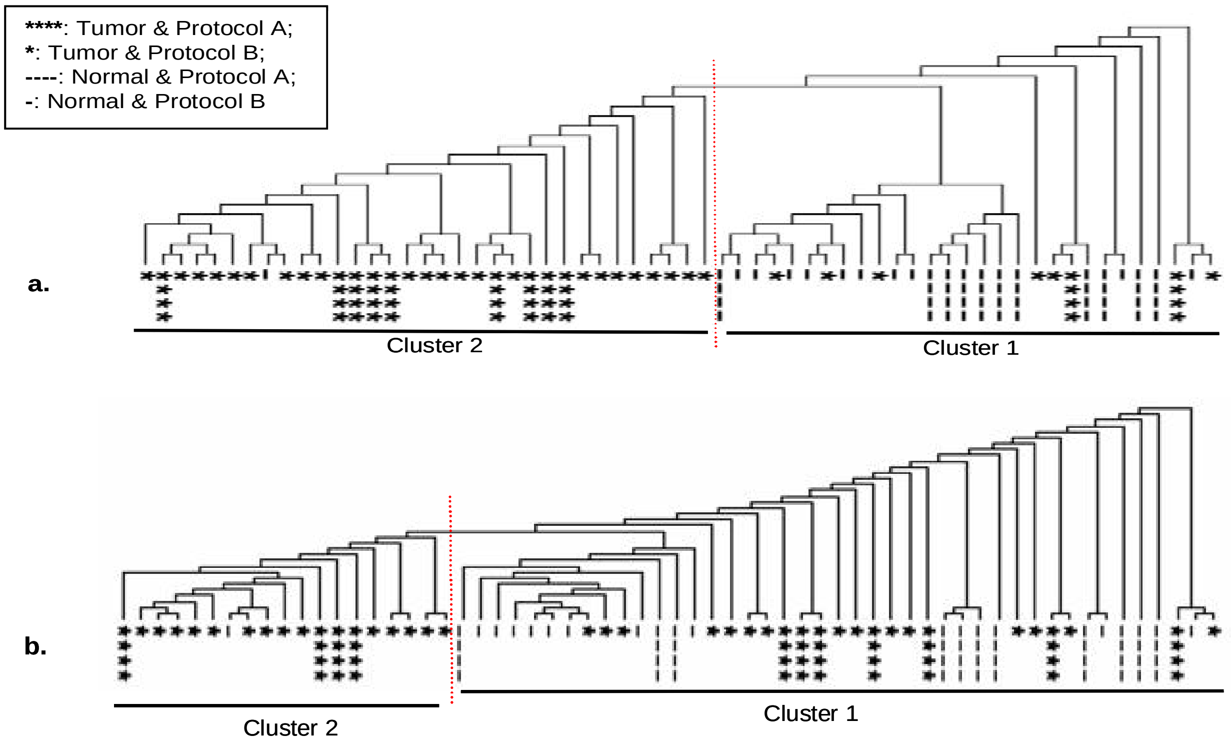

Figure 4. Clustering Dendrograms of 62 Samples with 2000 Genes as the Features by *a*, SMPC; *b*, MPC. (NJplot (Perrière *et al.*, 1996) is used to draw the dendrograms)

		Cluster 1	Cluster 2	Classification Rate		Cluster 1	Cluster 2	Classification Rate
SMPC	Normal	21	1	(21+32) / 62 = **85.48%**	Protocol A	13	9	(13+24) / 62 = **59.68%**
	Tumor	8	32		Protocol B	16	24	
MPC	Normal	21	1	(21+18) / 62 = **62.90%**	Protocol A	18	4	(18+18) / 62 = **58.06%**
	Tumor	22	18		Protocol B	25	18	

Table 1. Classification Rate of 62 Samples

CONCLUSIONS

We introduce the method of SMPC and its advantages over the deterministic algorithms in general and over the deterministic MPC in particular in clustering gene microarray data. SMPC generalizes MPC by involving stochastic processes and it can be reduced to MPC if we choose the particular kernel function (rectangular) and the particular bandwidth parameter (the minimum distance among distances between the target cluster and all the others) to estimate the probability. SMPC provides more reasonable strategy when deciding the merging pair: it assigns the merging probabilities to each cluster with respect to some target based on kernel functions, and it uses random switch technique to pick the cluster to merge given the probabilities. The closer the cluster is to the target, the higher the merging probability of the cluster, and the more chance that the cluster is selected.

Clustering results obtained by SMPC give more possible solutions, providing more points of view about the relationship among data, as shown in the example of the artificial data. SMPC is especially useful for identifying the relationship in the data where many distances are identical as in the protein interaction data. The analysis of the real colon cancer data with SMPC shows higher classification accuracy as compared to MPC.

DISCUSSIONS

In this paper, we apply the stochastic processes in conjunction with the MPC method due to its success in clustering parallel and spontaneous biological data; however, this strategy can be applied to many other deterministic clustering methods, such as hierarchical clustering and superparamagnetic clustering. Some theoretical issues (optimality of the bandwidth parameter *w* and comparison of different solutions from SMPC regarding the biological meaning, for example) will be addressed in future studies. Also, in preparation is the analysis of the real protein interaction data using SMPC, as in the set of 34 *S.cerevisiae* proteins characterized by Drees *et al.* (2001).

The undoing of agglomerating clusters is not addressed in this paper, but it can be implemented in a similar fashion. Future developments also include ensemble clustering, i.e. generating a large number of clustering solutions using SMPC (since it is stochastic), then using certain algorithm (such as a voting algorithm) to determine a final clustering result. Since this method is also index free, the clustering result is unbiased.

ACKNOWLEDGEMENTS

This work was supported by the NIH grant number P20 RR16469 from the INBRE program of National Center for Research Resource.

REFERENCES

1. Alon, U., Barkai, N., Notterman, D. A., Gish, K., Ybarra, S., Mack, D. and Levine, A. J. (1999) Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays, *PNAS*, 96**,** 6745–6750.
2. Arnau, V., Mars, S. and Marin, I. (2005) Iterative Cluster Analysis of Protein Interaction Data, *Bioinformatics*, vol. 21, pp. 364-378.
3. Ball, G. and Hall, D. (1967) A clustering technique for summarizing multivariate data, *Behaviorial Sciences*, 12(2), 153-155.

4. Dress, B. L., Sundin, B., BBrazeau, E., Caviston, J. P., Chen, G. C., Guo, W., Kozminske, K.G., Lau, M.W., Moskow, J. J., Tong, A., *et al.* (2001) A protein interaction map for cell polarity development, *J. Cell Biol.*, 154, 549-571.
5. Durbin, R., Eddy, S., Krogh, A. and Mitchison, G. (1998) Biological sequence analysis, Probabilistic models of proteins and nucleic acids, Cambridge University Press, New York.
6. Eisen, M. B., Spellman, P. T., Brown, P. O. and Botstein, D. (1998) Cluster analysis and display of genome-wide expression patterns, *Proc Natl Acad Sci, U S A*, 95, 14863-14868.
7. Everitt, B., Landau, S. and Leese, M. (2003) Cluster Alalysis, Oxford University Press, New York.
8. Felsenstein, J. (1989) PHYLIP (Phylogeny Inference Package), *Cladistics*, 5, 164-166.
9. Geng, H., Bastola, D. and Ali, H. (2004) A New Approach to Clustering Biological Data Using Message Passing. *Proceedings of IEEE Computer Society Bioinformatics Conference (CSB 2004)*, 493-494.
10. Getz, G., Levine, E. and Domany, E. (2000) Coupled two-way clustering analysis of gene microarray data, *PNAS*, 97, 12079-12084.
11. Haukins, D.M., Muller, M.W. and Krooden, J.A. (1982) cluster analysis, in Topics in Applied Multivariate Analysis, Cambridge University Press, Cambridge.
12. Herwig, R., Poustka, A. J., Meuller, C., Lehrach, H. and O'Brien, J. (1999) Large-scale clustering of cDNA-fingerprinting data, *Genome Research*, 9(11), 1093-1105.
13. Holter, M. S., Mitra, M., Maritan, A., Cieplak, M., Banavar, J. T. and Fedoroff, N. V. (2000) Fundamental Patterns Underlying Gene Expression Profiles: Simplicity from Complexity, *PNAS*, 97, 8409-8414.
14. Kaufman, L. and Rousseeuw, P.J. (1990) Finding Groups in Data. An Introduction to Cluster Analysis. Wiley-Interscience, New York.
15. Kohonen, T. (1997) Self-Organizing Maps, Springer, Berlin.
16. Lance, G. N. and Williams, W. T. (1967) A general theory of classification sorting strategies, *the computer journal*, 9, 373-380.
17. MacCuish, J., Nicolaou, C. and MacCuish, N. E. (2001) Ties in proximity and clustering compounds, *J. Chem. Inf. Comput. Sci.*, 41, 134–146.
18. Perrière, G. and Gouy, M. (1996) WWW-Query: An on-line retrieval system for biological sequence banks, *Biochimie*, 78, 364-369.
19. Phimister, B. *et al.* (1999) The Chipping Forecast , *Supplement to Nature Genetics* 21, 1-60.
20. Ross, S.M. (2002) Simulation, Academic Press, San Diego.
21. Russell, S. and Norvig, P. (2003) Artificial Intelligence, A Model Approach, Pearson Education, New Jersey.
22. Saitou, N. and Nei, M. (1987) The neighbor-joining method: a new method for reconstructing phylogenetic trees, *Mol. Biol. Evol.*, 4, 406-425.
23. Sharan, R. and Shamir, R. (2000) CLICK: A clustering algorithm with applications to gene expression analysis, *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology (ISMB)*, 307-316.
24. Takezaki, N. (1998) Tie trees generated by distance methods of phylogenetic reconstruction. *Mol. Biol. Evol.*, 15, 727–737.
25. Tamayo, P., Slonim, D., Mesirov, J., Zhu, Q., Kitareewan, S., Dmitrovsky, E., Lander, E. S. and Golub, T.R. (1999) Interpreting patterns of gene expression with self-organizing maps: Methods and application to hematopoietic differentiation, *PNAS*, 96, 2907-2912.