

# Bayesian Cluster Ensembles

Hongjun Wang<sup>1</sup>, Hanhuai Shan<sup>2\*</sup> and Arindam Banerjee<sup>2</sup>

<sup>1</sup>*Information Research Institute, Southwest Jiaotong University, Chengdu, Sichuan, 610031, China*

<sup>2</sup>*Department of Computer Science & Engineering, University of Minnesota, Twin Cities. Minneapolis, MN 55455*

Received 14 November 2009; revised 19 October 2010; accepted 27 October 2010

DOI:10.1002/sam.10098

Published online in Wiley Online Library (wileyonlinelibrary.com).

**Abstract:** Cluster ensembles provide a framework for combining multiple base clusterings of a dataset to generate a stable and robust consensus clustering. There are important variants of the basic cluster ensemble problem, notably including cluster ensembles with missing values, row- or column-distributed cluster ensembles. Existing cluster ensemble algorithms are applicable only to a small subset of these variants. In this paper, we propose Bayesian cluster ensemble (BCE), which is a mixed-membership model for learning cluster ensembles, and is applicable to all the primary variants of the problem. We propose a variational approximation based algorithm for learning Bayesian cluster ensembles. BCE is further generalized to deal with the case where the features of original data points are available, referred to as generalized BCE (GBCE). We compare BCE extensively with several other cluster ensemble algorithms, and demonstrate that BCE is not only versatile in terms of its applicability but also outperforms other algorithms in terms of stability and accuracy. Moreover, GBCE can have higher accuracy than BCE, especially with only a small number of available base clusterings. © 2011 Wiley Periodicals, Inc. *Statistical Analysis and Data Mining*, 2011

**Keywords:** cluster ensembles; Bayesian models

## 1. INTRODUCTION

Cluster ensembles provide a framework for combining multiple base clusterings of a dataset into a single consolidated clustering. Compared to individual clustering algorithms, cluster ensembles generate more robust and stable clustering results [1]. In principle, cluster ensembles can leverage distributed computing by calculating the base clusterings in an entirely distributed manner [2]. In addition, since cluster ensembles only need access to the base clustering results instead of the original data points, they provide a convenient approach to privacy preservation and knowledge reuse [2]. Such desirable aspects have made the study of cluster ensembles increasingly important in the context of data mining.

In addition to generating a consensus clustering from a complete set of base clusterings, it is highly desirable for cluster ensemble algorithms to have several additional properties suitable for real life applications. First, there may be missing values in the base clusterings. For example, in a customer segmentation application, while there are legacy

clusterings on old customers, there will be no clusterings on the new customers. Cluster ensemble algorithms should be able to build consensus clusters with such missing information on base clusterings. Second, there may be restrictions on bringing all the base clusterings to one place to run the cluster ensemble algorithm. Such restrictions may be due to the fact that the base clusterings are with different organizations and cannot be shared with each other. Cluster ensemble algorithms should be able to work with such ‘column-distributed’ base clusterings. Third, the data points themselves may be distributed over multiple locations; while it is possible to get a base clustering across the entire dataset by message passing, base clusterings for different parts of data will be in different locations, and there may be restrictions on bringing them together at one place. For example, for a customer segmentation application, different vendors may have different subsets of customers, and a base clustering on all the customers can be performed using privacy preserving clustering algorithms; however, the cluster assignments of the customer subsets for each vendor is private information which they will be unwilling to share directly for the purposes of forming a consensus clustering. Again, it will be desirable to have

*Correspondence to:* Hanhuai Shan (shan@cs.umn.edu)

cluster ensemble algorithms handle such ‘row-distributed’ base clusterings. Finally, in many real-world scenarios, features of original data points are available. These features could be the ones used for generating the base clusterings in the ensemble, or they could be the new information currently becoming available, such as some new purchasing records of a customer. In such a situation, a cluster ensemble algorithm which is able to combine both the base clustering results and data point’s features is expected to generate a better consensus clustering compared to using base clustering results alone.

Current cluster ensemble algorithms, such as the cluster-based similarity partitioning algorithm (CSPA) [2], hypergraph partitioning algorithm (HGPA) [2], or  $k$ -means based algorithms [3] are applicable to accomplish one or two of the above variants of the problem. However, none of them was designed to address all of the variants. In principle, the recently proposed mixture modeling approach to learning cluster ensembles [1] is applicable to the variants, but the details have not been reported in the literature. In this paper, we propose Bayesian cluster ensembles (BCE), which can solve the basic cluster ensemble problem using a Bayesian approach, that is, by effectively maintaining a distribution over all possible consensus clusterings. It also seamlessly generalizes to all the important variants discussed above. Similar to the mixture modeling approach, BCE treats all base clustering results for each data point as a vector with a discrete value on each dimension, and learns a mixed-membership model from such a representation. In addition, we extend BCE to generalized BCE (GBCE), which learns a consensus clustering from both the base clusterings and feature vectors of original data points. Extensive empirical evaluation demonstrates that BCE is not only versatile in terms of its applicability but also mostly outperforms the other cluster ensemble algorithms in terms of stability and accuracy. Moreover GBCE can have higher accuracy than BCE, especially when there are only a small number of available base clusterings.

The rest of the paper is organized as follows. In Section 2, we give a problem definition. Section 3 presents the related work in cluster ensembles. The model for BCE is proposed in Section 4, and a variational inference algorithm is discussed in Section 5. Section 6 proposes GBCE. We report experimental results in Section 7, and conclude in Section 8.

## 2. PROBLEM FORMULATION

Given  $N$  data points  $O = \{\mathbf{o}_i, [i]_1^N\}$  ( $[i]_1^N \equiv i = 1, \dots, N$ ) and  $M$  base clustering algorithms  $C = \{c_j, [j]_1^M\}$ , we get  $M$  base clusterings of the data points, one from each algorithm. The only requirement from a base clustering

algorithm is that it generates a cluster assignment or id for each of the  $N$  data points  $\{\mathbf{o}_i, [i]_1^N\}$ . The number of clusters generated by different base clustering algorithms may be different. We denote the number of clusters generated from  $c_j$  by  $k_j$ , so that the cluster ids assigned by  $c_j$  range from 1 to  $k_j$ . If  $\lambda_{ij} \in \{1, \dots, k_j\}$  denotes the cluster id assigned to  $\mathbf{o}_i$  by  $c_j$ , the base clustering algorithm  $c_j$  gives a clustering of the entire dataset, given by

$$\lambda_j = \{\lambda_{ij}, [i]_1^N\} = \{c_j(\mathbf{o}_i), [i]_1^N\}.$$

The results from  $M$  base clustering algorithms can be stacked together to form an  $(N \times M)$  matrix  $B$ , whose  $j$ th column is  $\lambda_j$ , as shown in panel (a) of Fig. 1. The matrix can be viewed from another perspective: Each row  $\mathbf{x}_i$  of the matrix, that is, all base clustering results for  $\mathbf{o}_i$ , gives a new vector representation for the data point  $\mathbf{o}_i$  (panel (b) of Fig. 1). In particular,

$$\mathbf{x}_i = \{x_{ij}, [j]_1^M\} = \{c_j(\mathbf{o}_i), [j]_1^M\}.$$

Given the base clustering matrix  $B$ , the cluster ensemble problem is to combine the  $M$  base clustering results for  $N$  data points to generate a consensus clustering, which should be more accurate, robust, and stable than the individual base clusterings. The traditional approach to process the base clustering results is ‘column-wise’ (panel (a) of Fig 1), that is, we consider  $B$  as a set of  $M$  columns of base clustering results  $\{\lambda_j, [j]_1^M\}$ , and we try to find out the consensus clustering  $\lambda^*$ . The disadvantage of the ‘column-wise’ perspective is that it needs to find out the correspondence between different base clusters generated by different algorithms. For example, in panel (a) of Fig. 1, we need to know ‘1’ in the first column corresponds to ‘1’ or ‘2’ or ‘3’ in the second column. The cluster correspondence problem is hard to solve efficiently, and the complexity increases especially when different base clustering algorithms generate different numbers of clusters [1].

A simpler approach to cluster ensemble problem, which is what we use in this paper, is to read the matrix  $B$  in

	$\lambda_1$	$\lambda_2$	...	$\lambda_M$		$\lambda_1$	$\lambda_2$	...	$\lambda_M$
$\mathbf{x}_1$	2	1	...	2		2	1	...	2
$\mathbf{x}_2$	1	3	...	3		1	3	...	3
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$		$\vdots$	$\vdots$	$\ddots$	$\vdots$
$\mathbf{x}_N$	3	2	...	6		3	2	...	6

Fig. 1 Two ways of processing base clustering results for cluster ensemble.

a ‘row-wise’ (panel (b) of Fig 1) way. All base clustering results for a data point  $\mathbf{o}_i$  can be considered as a vector  $\mathbf{x}_i$  with discrete values on each dimension [1], and we consider base clustering matrix  $B$  as a set of  $N$  rows of  $M$ -dimensional vectors  $\{\mathbf{x}_i, [i]_1^N\}$ . From this perspective, the cluster ensemble problem becomes finding a clustering  $\lambda^*$  for  $\{\mathbf{x}_i, [i]_1^N\}$ , where  $\lambda^*$  is a consensus clustering over all base clusterings. Further, by considering the cluster ensemble problem from this perspective, we naturally avoid cluster correspondence problem, because for each  $\mathbf{x}_i$ ,  $\lambda_1$  and  $\lambda_2$  are just two features, they are conditionally independent in the naive Bayes setting for clustering.

While the basic cluster ensemble framework assumes all base clustering results for all data points are available in one place to perform the analysis, real-life applications often need variants of the basic setting. In this paper, we discuss four important variants: missing value cluster ensembles, row- and column-distributed cluster ensembles, and cluster ensemble with original data points.

### 2.1. Missing Value Cluster Ensembles

When several base clustering results are missing for several data points, we have a missing value cluster ensemble problem. Such a problem appears due to various reasons. For example, if there are new data points added to the dataset after running clustering algorithm  $c_j$ , these new data points will not have base clustering results corresponding to  $c_j$ . In missing value cluster ensemble, instead of dealing with a full base clustering matrix  $B$ , we are dealing with a matrix with missing entries.

### 2.2. Row-Distributed Cluster Ensembles

For row-distributed cluster ensembles, base clustering results of different data points (rows) are at different locations. The corresponding real-life scenario is that different subsets of the original dataset are owned by different organizations, or cannot be put together in one place due to size, communication, or privacy constraints. While distributed base clustering algorithms, such as distributed privacy preserving  $k$ -means [4], can be run on the subsets to generate base clustering results, due to the restrictions on sharing, the results on different subsets cannot be transmitted to a central location for analysis. Therefore, it is desirable to learn a consensus clustering in a row-distributed manner.

### 2.3. Column-Distributed Cluster Ensembles

For column-distributed cluster ensemble, different base clustering results of all data points are at different

locations. The corresponding real-life scenario is that separate organizations have different base clusterings on the same set of data points, for example, different e-commerce vendors having customer segmentations on the same customer base. The base clusterings cannot be shared with others due to privacy concerns, but each organization has an incentive to get a more robust consensus clustering. In such a case, the cluster ensemble problem have to be solved in a column-distributed way.

### 2.4. Cluster Ensemble with Original Data Points

In many real-life scenarios, not only the base clustering results but also the features of original data points are available. For example, a company may have both the customer segmentations and their purchasing records. The features of original data points could be the ones used to generate the base clustering results, for example, the purchasing records used to generate the existent customer segmentations. The features could also be the new information currently become available, for example, new purchasing records of customers. In such cases, we may lose useful information by running cluster ensemble algorithms on base clustering results only. Meanwhile, if the base clustering algorithms do not perform very well, a combination of them usually fails to yield a good consensus clustering. Therefore, a cluster ensemble algorithm which can take both the base clustering results and original data points is expected to generate a better consensus clustering.

## 3. RELATED WORK

In this section, we give a brief overview of cluster ensemble algorithms. There are three main classes of algorithms: graph-based models, matrix-based models, and probabilistic models.

### 3.1. Graph-Based Models

The most popular algorithms for cluster ensemble are graph-based models [2,5–7]. The main idea of this class of algorithms is to convert the results of base clusterings to a hypergraph or a graph and then use graph partitioning algorithms to obtain ensemble clusters.

Strehl and Ghosh [2] present three graph-based cluster ensemble algorithms: CSPA [2] induces a graph from a co-association matrix, and the graph is partitioned by the METIS algorithm [8] to obtain final clusters. In addition, HGPA [2] represents each cluster and corresponding objects by a hyperedge and nodes, respectively, and then uses minimal cut algorithm HMTIS [9] for partitioning.

Further, hyperedge collapsing operations are used in a meta-clustering algorithm (MCLA) [2] which determines a soft cluster membership for each object.

Fern and Brodley [5] propose a bipartite graph partitioning algorithm. It solves cluster ensemble by reducing it to a graph partitioning problem and introduces a new reduction method that constructs a bipartite graph from the base clusterings. The graph models consider both objects and clusters of the ensemble as vertices simultaneously.

Al-Razgan and Domeniconi [6] propose a weighted bipartite partitioning algorithm (WBPA), which maps the problem of finding a consensus partition to bipartite graph partitioning.

### 3.2. Matrix-Based Models

The second class of algorithms are matrix-based models [10–13]. The main idea of this category is converting base clustering matrix to another matrix such as co-association matrix, consensus matrix, or non-negative matrix, and using matrix operations to get the results of cluster ensemble.

Fred and Jain [10] map various base clustering results to a co-association matrix, where each entry represents the strength of association between objects, based on the co-occurrence of two objects in a same cluster. A voting algorithm is applied to the co-association matrix to obtain the final result. Clusters are formed from the co-association matrix by collecting the objects whose co-association values exceed the threshold.

Kellam *et al.* [12] combine results of base clusterings through a co-association matrix, which is an agreement matrix with each cell containing the number of agreements among the base clustering methods. The co-association matrix is used to find the clusters with the highest value of support based on object co-occurrences. As a result, only a set of so-called ‘robust clusters’ are produced.

Monti *et al.* [13] define a consensus matrix for representing and quantifying the agreement among the results of base clusterings. For each pair of objects, the matrix stores the proportion of clustering runs in which two objects are clustered together.

Li *et al.* [11] illustrate that the problem of cluster ensemble can be formulated under the framework of non-negative matrix factorization (NMF), which refers to the problem of factorizing a given non-negative data matrix  $X$  into two matrix factors that is,  $X \approx AB$ , under the constraint of  $A$  and  $B$  to be non-negative matrices.

### 3.3. Probabilistic Models

The third class of cluster ensemble algorithms are based on probabilistic models [1]. The algorithms take

advantage of statistic properties of base clusterings results to achieve a consensus clustering. Topchy *et al.* [1] consider a representation of multiple clusterings as a set of new attributes characterizing the data items, and a mixture model (MM) offers a probabilistic model of consensus using a finite mixture of multinomial distributions in the space of base clusterings. A consensus result is found as a solution to the corresponding maximum likelihood problem using expectation maximization (EM) algorithm.

## 4. BAYESIAN CLUSTER ENSEMBLES

In this section, we propose a novel BCE model. The main idea is as follows: Given a base clustering matrix  $B = \{\mathbf{x}_i, [i]_1^N\}$  for  $N$  data points, we assume there exists a Bayesian graphical model generating  $B$ . In particular, we assume that each vector  $\mathbf{x}_i$  has an underlying mixed-membership to different consensus clusters. Let  $\theta_i$  denote the latent mixed-membership vector for  $\mathbf{x}_i$ ; if there are  $k$  consensus clusters,  $\theta_i$  is a discrete distribution over the  $k$  clusters. From the generative model perspective, we assume that  $\theta_i$  is sampled from a Dirichlet distribution, with parameter  $\alpha$ , and the consensus cluster  $h$ ,  $[h]_1^k$  for each  $x_{ij}$  of  $[j]_1^M$  is sampled from  $\theta_i$  separately. Further, each latent consensus cluster  $h$ , has a discrete distribution  $\beta_{hj}$  over the cluster ids  $\{1, \dots, k_j\}$  for the  $j$ th base clustering result of each  $\mathbf{x}_i$ . Thus, if  $x_{ij}$  truly belongs to consensus cluster  $h$ ,  $x_{ij} = r \in \{1, \dots, k_j\}$  will be determined by the discrete probability distribution  $\beta_{hj}(r) = p(x_{ij}|\beta_{hj})$ , where  $\beta_{hj}(r) \geq 0$ ,  $\sum_{r=1}^{k_j} \beta_{hj}(r) = 1$ . The full generative process for each  $\mathbf{x}_i$  is assumed to be as follows (Fig. 2):

1. Choose  $\theta_i \sim \text{Dirichlet}(\alpha)$ .
2. For the  $j$ th base clustering:
  - (a) Choose a component  $z_{ij} = h \sim \text{discrete}(\theta_i)$ ;
  - (b) Choose the base clustering result  $x_{ij} \sim \text{discrete}(\beta_{hj})$ .

Thus, the model contains the model parameters  $(\alpha, \beta)$ , where  $\beta = \{\beta_{hj}, [h]_1^k, [j]_1^M\}$ , the latent variables  $(\theta_i, z_{ij})$  and the actual observations  $\{x_{ij}, [i]_1^N, [j]_1^M\}$ . BCE can be viewed as a special case of mixed-membership naive Bayes models [14,15] by choosing a discrete distribution as the generative model. Further, BCE is closely related to LDA [16], although the models are applicable to different types of data.

Given the model parameters  $\alpha$  and  $\beta$ , the joint distribution of latent and observed variables  $\{\mathbf{x}_i, \mathbf{z}_i, \theta_i\}$  is

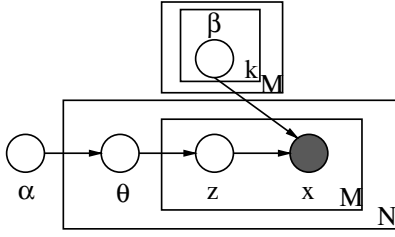


Fig. 2 Graphical model for BCE.

given by:

$$p(\mathbf{x}_i, \boldsymbol{\theta}_i, \mathbf{z}_i | \alpha, \beta) = p(\boldsymbol{\theta}_i | \alpha) \prod_{j=1, \exists x_{ij}}^M p(z_{ij} = h | \boldsymbol{\theta}_i) p(x_{ij} | \beta_{hj}),$$

where  $\exists x_{ij}$  denotes that there exists a  $j$ th base clustering result for  $\mathbf{x}_i$ , so the product is only over the existing base clustering results. By integrating over the latent variables  $\{\mathbf{z}_i, \boldsymbol{\theta}_i\}$ , the marginal probability for each  $\mathbf{x}_i$  is given by:

$$p(\mathbf{x}_i | \alpha, \beta) = \int_{\boldsymbol{\theta}_i} p(\boldsymbol{\theta}_i | \alpha) \prod_{j=1, \exists x_{ij}}^M \sum_h p(z_{ij} = h | \boldsymbol{\theta}_i) p(x_{ij} | \beta_{hj}) d\boldsymbol{\theta}_i. \quad (1)$$

BCE could be considered as a generalization of mixture models [1] to Bayesian models on cluster ensemble problem. In MMs, for each  $\mathbf{x}_i$  we pick a  $\mathbf{z}_i$ , which may take  $[0, 0, 1]$ ,  $[0, 1, 0]$ , or  $[1, 0, 0]$  in a three cluster problem, that is, there are only three possible values for the consensus clustering in the generative process. In comparison, in BCE, for each  $\mathbf{x}_i$  we pick a  $\boldsymbol{\theta}_i$ , which could be any valid discrete distribution, that is, in this case, it could be any three-dimensional vector with each dimension larger than 0 and the summation of three dimensions equal to 1. Also, we keep a Dirichlet distribution over all possible  $\boldsymbol{\theta}_i$ s. Such a scheme of BCE is better than that of MMs due to the two reasons: (i) The membership vector of BCE has a much larger set of choices than MMs. (ii) BCE allows mixed membership (a membership to multiple consensus clusters) in the generative process, while MMs only allow a sole membership (a membership to only one consensus cluster). Therefore, BCE is more flexible than mixture model based cluster ensembles.

## 5. VARIATIONAL INFERENCE FOR BCE

We have assumed a generative process for the base clustering matrix  $B = \{\mathbf{x}_i, [i]_1^N\}$  in Section 4. Given the observable matrix  $B$ , our final goal is to estimate the mixed-membership  $\{\boldsymbol{\theta}_i, [i]_1^N\}$  of each object to the consensus clusters. Since the model parameters  $\alpha$  and  $\beta$  are

unknown, we have to also estimate the model parameters such that the log-likelihood of observing the base clustering matrix  $B$  is maximized. EM algorithms are typically used for such parameter estimation problems by alternating between calculating the posterior over latent variables and updating the model parameters until convergence. However, the posterior distribution

$$p(\boldsymbol{\theta}_i, \mathbf{z}_i | \mathbf{x}_i, \alpha, \beta) = \frac{p(\boldsymbol{\theta}_i, \mathbf{z}_i, \mathbf{x}_i | \alpha, \beta)}{p(\mathbf{x}_i | \alpha, \beta)} \quad (2)$$

cannot be calculated in a closed form since the denominator (partition function)  $p(\mathbf{x}_i | \alpha, \beta)$  as an expansion of Eq. (1) is given by

$$p(\mathbf{x}_i | \alpha, \beta) = \int \frac{\Gamma(\sum_h \alpha_h)}{\prod_h \Gamma(\alpha_h)} \left( \prod_{h=1}^k \theta_{ih}^{\alpha_h - 1} \right) \prod_{j=1}^M \sum_{h=1}^k \theta_{ih} \prod_{r=1}^{k_j} \beta_{hj}(r)^{\mathbb{1}(r|i,j)} d\boldsymbol{\theta}_i,$$

where  $\mathbb{1}(r|i,j)$  is an indicator taking value 1 if the  $j$ th base clustering assigns  $\mathbf{o}_i$  to base cluster  $r$  and 0 otherwise,  $\beta_{hj}(r)$  is the  $r$ th component of the discrete distribution  $\beta_{hj}$  for the  $h$ th consensus cluster and the  $j$ th base clustering. The coupling between  $\theta$  and  $\beta$  in the summation over the latent variable  $z$  makes the computation intractable [16]. There are two main classes of approximation algorithms to address such problems: one is variational inference, and the other is Gibbs sampling. In our paper, we present the variational inference method.

### 5.1. Variational Inference

Since it is intractable to calculate the true posterior in Eq. (2) directly, in variational inference, we introduce a family of distributions as an approximation of the posterior distribution over latent variables to get a tractable lower bound of the log-likelihood  $\log(p(\mathbf{x}_i | \alpha, \beta))$ . We maximize this lower bound to update the parameter estimation. In particular, following [14,16], we introduce a family of variational distributions as

$$q(\boldsymbol{\theta}_i, \mathbf{z}_i | \gamma_i, \phi_i) = q(\boldsymbol{\theta}_i | \gamma_i) \prod_{j=1}^M q(z_{ij} | \phi_{ij}) \quad (3)$$

as an approximation of  $p(\boldsymbol{\theta}_i, \mathbf{z}_i | \alpha, \beta, \mathbf{x}_i)$  in Eq. (2), where  $\gamma_i$  is a Dirichlet distribution parameter, and  $\phi_i = \{\phi_{ij}, [j]_1^M\}$  are discrete distribution parameters. We introduce such an approximating distribution for each  $\mathbf{x}_i, [i]_1^N$ . Now, using Jensen's inequality [17], we can obtain a lower bound

$L(\alpha, \beta; \phi_i, \gamma_i)$  to  $\log p(\mathbf{x}_i|\alpha, \beta)$  given by:

$$L(\alpha, \beta; \phi_i, \gamma_i) = E_q[\log p(\boldsymbol{\theta}_i, \mathbf{z}_i|\alpha, \beta)] + H(q(\boldsymbol{\theta}_i, \mathbf{z}_i|\gamma_i, \phi_i)),$$

where  $H(\cdot)$  denotes the Shannon entropy. Assuming each row  $\mathbf{x}_i$  of the matrix  $B$  to be statistically independent given the parameters  $(\alpha, \beta)$ , the log-likelihood of observing the matrix  $B$  is simply

$$\log p(B|\alpha, \beta) = \sum_{i=1}^N \log p(\mathbf{x}_i|\alpha, \beta) \geq \sum_{i=1}^N L(\alpha, \beta; \phi_i, \gamma_i). \quad (4)$$

For a fixed set of model parameters  $(\alpha, \beta)$ , maximizing the lower bound with respect to the free variational parameters  $(\gamma_i, \phi_i)$  for each  $\mathbf{x}_i$ ,  $[i]_1^N$  gives us the best lower bound from this family of approximations. A direct calculation leads to the following set of update equations for the variational maximization:

$$\phi_{ijh} \propto \exp\left(\Psi(\gamma_{ih}) - \Psi\left(\sum_{h'=1}^k \gamma_{ih'}\right)\right) \quad (5)$$

$$+ \sum_{r=1}^{k_j} \mathbb{1}(r|i, j) \log \beta_{hj}(r),$$

$$\gamma_{ih} = \alpha_h + \sum_{j=1, \exists \mathbf{x}_{ij}}^M \phi_{ijh}, \quad (6)$$

where  $[i]_1^N$ ,  $[j]_1^M$ ,  $[h]_1^k$ ,  $\phi_{ijh}$  is the  $h$ th component of the variational discrete distribution  $\phi_{ij}$  for  $z_{ij}$ , and  $\gamma_{ih}$  is the  $h$ th component of the variational Dirichlet distribution  $\gamma_i$  for  $\boldsymbol{\theta}_i$ .

For a given set of variational parameters  $(\gamma_i, \phi_i)$ ,  $[i]_1^N$ , the lower bound given in Eq. (4) is maximized by the point estimate for  $\beta$ :

$$\beta_{hj}(r) \propto \sum_{i=1}^N \phi_{ijh} \mathbb{1}(r|i, j), \quad (7)$$

where  $[h]_1^k$ ,  $[j]_1^M$ ,  $[r]_1^{k_j}$ . The Dirichlet parameter  $\alpha$  can be estimated via Newton–Raphson updates as in LDA [16]. In particular, the update equation for  $\alpha_h$  is given by

$$\alpha'_h = \alpha_h - \frac{g_h - c}{l_h}, \quad (8)$$

with

$$g_h = N \left( \Psi\left(\sum_{h'=1}^k \alpha_{h'}\right) - \Psi(\alpha_h) \right) + \sum_{i=1}^N \left( \Psi(\gamma_{ih}) - \Psi\left(\sum_{h'=1}^k \gamma_{ih'}\right) \right),$$

$$l_h = -N \Psi'(\alpha_h),$$

$$c = \frac{\sum_{h=1}^k g_h / l_h}{v^{-1} + \sum_{h=1}^k l_h^{-1}},$$

$$v = N \Psi'\left(\sum_{h=1}^k \alpha_h\right),$$

where  $\Psi$  is the digamma function, that is, the first derivative of the log Gamma function.

## 5.2. Variational EM Algorithms

Given the updating equations for variational parameters and model parameters, we can use a variational EM algorithm to find the best-fit model  $(\alpha^*, \beta^*)$ . Starting from an initial guess  $(\alpha^{(0)}, \beta^{(0)})$ , the EM algorithm alternates between two steps until convergence:

1. E-Step: Given  $(\alpha^{(t-1)}, \beta^{(t-1)})$ , for each  $\mathbf{x}_i$ , find the best variational parameters:

$$(\phi_i^{(t)}, \gamma_i^{(t)}) = \underset{(\phi_i, \gamma_i)}{\operatorname{argmax}} L(\alpha^{(t)}, \beta^{(t)}; \phi_i, \gamma_i).$$

$L(\alpha, \beta, \phi_i^{(t)}, \gamma_i^{(t)})$  serves as a lower bound function to  $\log p(\mathbf{x}_i|\alpha, \beta)$ .

2. M-Step: Maximize the aggregate lower bound with respect to  $(\alpha, \beta)$  to obtain an improved parameter estimate:

$$(\alpha^{(t)}, \beta^{(t)}) = \underset{(\alpha, \beta)}{\operatorname{argmax}} \sum_{i=1}^N L(\alpha, \beta; \phi_i^{(t)}, \gamma_i^{(t)}).$$

After  $(t - 1)$  iterations, the value of the lower bound function is  $L(\alpha^{(t-1)}, \beta^{(t-1)}, \phi_i^{(t-1)}, \gamma_i^{(t-1)})$ . In the  $t$ th iteration,

$$\sum_{i=1}^N L(\alpha^{(t-1)}, \beta^{(t-1)}, \phi_i^{(t-1)}, \gamma_i^{(t-1)}) \leq \sum_{i=1}^N L(\alpha^{(t-1)}, \beta^{(t-1)}, \phi_i^{(t)}, \gamma_i^{(t)}) \quad (9)$$

$$\leq \sum_{i=1}^N L(\alpha^{(t)}, \beta^{(t)}, \phi_i^{(t)}, \gamma_i^{(t)}). \quad (10)$$

The first inequality holds because in the E-step, Eq. (9) is the maximum of  $L(\alpha^{(t-1)}, \beta^{(t-1)}, \phi_i, \gamma_i)$ , and the second inequality holds because in the M-step, Eq. (10) is the maximum of  $L(\alpha, \beta, \phi_i^{(t)}, \gamma_i^{(t)})$ . Therefore, the objective function is nondecreasing until convergence [17].

For computational complexity, since we only need to calculate  $\Psi(\gamma_{ih}) - \Psi(\sum_{h'=1}^k \gamma_{ih'})$  once for all  $\phi_{ijh}, [j]_1^M$ , the complexity for updating  $\phi$  in each E-step is then  $O((Nk^2 + NMku)t_E)$ , where  $u = \max\{k_j, [j]_1^M\}$  and  $t_E$  is the number of iterations inside each E-step. Also, the time for updating  $\gamma$  is  $O(NMkt_E)$ . In the M-step, the complexity for updating  $\beta$  is  $O(NMku)$ .  $\alpha$  is updated using Newton update and the time needed is  $O(kNt_\alpha)$ , where  $t_\alpha$  is the number of iterations in Newton updates. Compared to MM based cluster ensemble algorithm [1], BCE is computationally more expensive, since it has iterations over Eqs. (5) and (6) inside the E-step, while [1] uses a direct EM algorithm which has the E-step in a closed form. However, as we show in the experiments, BCE achieves significantly better performance than MMs.

### 5.2.1. Row-distributed EM algorithm

In row-distributed cluster ensemble, the object set  $O$  is partitioned into  $P$  parts  $\{O_{(1)}, O_{(2)}, \dots, O_{(P)}\}$  and different parts are assumed to be at different locations. We further assume that a set of distributed base clustering algorithms have been used to obtain the base clustering results  $\{B_{(1)}, B_{(2)}, \dots, B_{(P)}\}$ . Now, we outline a row-distributed variant of the variational inference algorithm. At each iteration  $t$ , given the initialization of model parameters  $(\alpha^{(t-1)}, \beta^{(t-1)})$ , row-distributed variational EM for BCE proceeds as follows:

1. For each partition  $\{B_{(p)}, [p]_1^P\}$ , we obtain variational parameters  $(\phi_{(p)}, \gamma_{(p)})$  following Eqs. (5) and (6), where  $\phi_{(p)} = \{\phi_i | \mathbf{x}_i \in B_{(p)}\}$  and  $\gamma_{(p)} = \{\gamma_i | \mathbf{x}_i \in B_{(p)}\}$ .
2. To update  $\beta$  following Eq. (7), we can write the right term of Eq. (7) as

$$\sum_{\mathbf{x}_i \in B_{(1)}} \phi_{ijh} \mathbb{1}(r|i, j) + \dots + \sum_{\mathbf{x}_i \in B_{(P)}} \phi_{ijh} \mathbb{1}(r|i, j).$$

Each part in the summation corresponds to one partition of  $B$ . To update  $\beta_{hj}(r)$ , first,  $\Delta_{(p)} = \sum_{\mathbf{x}_i \in B_{(p)}} \phi_{ijh} \mathbb{1}(r|i, j)$  is calculated for each  $B_p$ . Second, for each  $B_{(p)} (p \in [2, P])$ , we take  $\sum_{q=1}^{p-1} \Delta_{(q)}$  from  $B_{(p-1)}$ , generate  $\sum_{q=1}^p \Delta_{(q)}$  by adding  $\Delta_{(p)}$  to the summation, and pass it to  $B_{(p+1)}$ . Finally, after passing through all partitions, we have the

summation as the right term of Eq. (7) to update  $\beta_{hj}(r)$  after normalization.

3. Updating  $\alpha$  is a little tricky since it does not have a closed form solution. However, we notice that the update Eq. (8) for  $\alpha'_h$  only depends on two variables:  $\alpha_h$  and  $\{\gamma_i, [i]_1^N\}$ .  $\alpha_h$  can be obtained from the last iteration of Newton–Raphson algorithm. Regarding  $\gamma$ , we only need to know  $\sum_{i=1}^N \Psi(\gamma_{ih})$  and  $\sum_{i=1}^N \Psi(\sum_h \gamma_{ih})$  for  $g$  in Eq. (8). We use a same strategy as for updating  $\beta$ : First we calculate  $\Lambda_p = \sum_{\mathbf{x}_i \in B_{(p)}} \Psi(\gamma_{ih})$  and  $\Omega_p = \sum_{\mathbf{x}_i \in B_{(p)}} \Psi(\sum_h \gamma_{ih})$  on each partition. Second, for each  $B_{(p)} (p \in [2, P])$ , we take  $\sum_{q=1}^{p-1} \Lambda_q$  and  $\sum_{q=1}^{p-1} \Omega_q$  from  $B_{(p-1)}$ , generate  $\sum_{q=1}^p \Lambda_q$  and  $\sum_{q=1}^p \Omega_q$  by adding  $\Lambda_p$  and  $\Omega_p$  to the summations respectively, and pass them to  $B_{(p+1)}$ . Finally, after going through all partitions, we have the result for  $\sum_{i=1}^N (\Psi(\gamma_{ih}) - \Psi(\sum_h \gamma_{ih}))$ , so we can update  $\alpha'_h$  following Eq. (8). For each iteration of Newton–Raphson algorithm, we need to pass the summations through all partitions once.

By the end of the  $t$ th iteration, we have the updated model parameters  $(\alpha^{(t)}, \beta^{(t)})$ , which are used as the initialization for the  $(t+1)$ th iteration. The algorithm is guaranteed to converge since it is essentially the same with the EM for the general case, except that it works in a row-distributed way. By running EM distributedly, neither  $\{O_{(p)}, [p]_1^P\}$  nor  $\{B_{(p)}, [p]_1^P\}$  is passed around different individuals, but only the intermediate summations; in this sense, we achieve privacy preservation.

As we have noticed, updating  $\alpha$  is very expensive because it needs to pass the summations over all partitions for each Newton–Raphson iteration, which is practically infeasible for a dataset with a large number of partitions. Therefore, we next give a heuristic row-distributed EM, which does not have a theoretical guarantee for convergence, but worked well in practice in our experiments.

At each iteration  $t$ , given the initialization of model parameters  $(\alpha_{(1)}^{(t-1)}, \beta_{(1)}^{(t-1)})$ , heuristic row-distributed variational EM for BCE proceeds as follows:

1. For the first partition  $B_{(1)}$ , given  $(\alpha_{(1)}^{(t-1)}, \beta_{(1)}^{(t-1)})$ , we obtain variational parameters  $(\phi_{(1)}, \gamma_{(1)})$  following Eqs. (5) and (6). Also, we update  $(\alpha_{(1)}, \beta_{(1)})$  to get  $(\alpha_{(1)}^{(t)}, \beta_{(1)}^{(t)})$  following Eqs. (8) and (7) respectively.
2. For the  $p$ th partition  $B_{(p)}$ , we initialize  $(\alpha_{(p)}, \beta_{(p)})$  with  $(\alpha_{(p-1)}^{(t)}, \beta_{(p-1)}^{(t)})$  and obtain  $(\phi_{(p)}, \gamma_{(p)})$  following Eqs. (5) and (6). We update  $(\alpha_{(p)}^{(t)}, \beta_{(p)}^{(t)})$  and pass them to the  $(p+1)$ th partition.

After going over all partitions, we are done with the  $t$ th iteration; the iterations are repeated until convergence. The initialization for  $(\alpha_{(1)}^{(1)}, \beta_{(1)}^{(1)})$  in the first iteration could be picked by random or by using some heuristics, and the initializations for  $(\alpha_{(1)}, \beta_{(1)})$  in the  $t$ th iteration are from  $(\alpha_{(P)}^{(t-1)}, \beta_{(P)}^{(t-1)})$ . The iterations run till the net change in the lower bound value is below a threshold, or when a pre-fixed number of iterations reached.

### 5.2.2. Column-distributed EM algorithm

For column-distributed cluster ensemble, we design a client-server style algorithm, where each client maintains one base clustering, and the server gathers partial results from the clients and performs further processing. While we assume that there are  $M$  different clients, one can always work with a smaller number of clients by splitting the columns among the available clients. Given the initialization for model parameters  $(\alpha^{(t)}, \beta^{(t)})$ , where  $(\alpha^{(t)}, \beta_{\cdot j}^{(t)})$  is made available to the  $j$ th client, the column-distributed cluster ensemble at iteration  $t$  proceeds as follows:

1. E-step  $j$ th client: Given  $x_{ij}$  and  $\beta_{\cdot j}^{(t)}$  for  $[i]_1^N$ , the  $j$ th client calculates  $\sum_{r=1}^{k_j} \mathbb{1}(r|i, j) \log \beta_{hj}^{(t)}(r)$  for  $[i]_1^N$ ,  $[h]_1^k$  and passes the results to the E-step server.
2. E-step server: Given  $\sum_{r=1}^{k_j} \mathbb{1}(r|i, j) \log \beta_{hj}^{(t)}(r)$  from the clients, for  $[i]_1^N$ ,  $[j]_1^M$ ,  $[h]_1^k$ , the server calculates variational parameters  $\{\phi_{ijh}, [i]_1^N, [j]_1^M, [h]_1^k\}$  following Eq. (5). Given  $\alpha^{(t)}$  and  $\{\phi_{ijh}, [i]_1^N, [j]_1^M, [h]_1^k\}$ , the server updates  $\{\gamma_{ih}, [i]_1^N, [h]_1^k\}$  following Eq. (6). The parameters  $\{\phi_{ijh}, [i]_1^N, [h]_1^k\}$  are passed to the M-step  $j$ th client and  $\{\gamma_{ih}, [i]_1^N, [h]_1^k\}$  are passed to the M-step server.
3. M-step  $j$ th client: Given  $x_{ij}$  and  $\phi_{ijh}$  for  $[i]_1^N$ ,  $[h]_1^k$ ,  $\beta_{\cdot j}^{(t+1)}(\cdot)$  is updated following Eq. (7) and passed to E-step server for the  $(t+1)$ th iteration.
4. M-step server: Given  $\alpha^{(t)}$  and  $\gamma_{ih}$  for  $[i]_1^N$ ,  $[h]_1^k$ ,  $\alpha^{(t+1)}$  is updated following Eq. (8) and passed to E-step server for the next step.

The initialization  $(\alpha^{(0)}, \beta^{(0)})$  is chosen at the beginning of the first iteration. In iteration  $t$ ,  $(\alpha^{(t)}, \beta^{(t)})$  are initialized by  $(\alpha^{(t-1)}, \beta^{(t-1)})$ , that is, the results of the  $(t-1)$ th iteration. The algorithm is guaranteed to converge because it is essentially the same as the EM algorithm for general cluster ensembles except that it is running in a column-distributed way. The algorithm is expected to be more efficient than the general cluster ensemble if we ignore the communication overhead. In addition,  $j$ th client/server only has access

to the  $j$ th base clustering results. The communication is only for the parameters and intermediate results, instead of base clusterings. Therefore, privacy preservation is also achieved.

In BCE, the most computationally expensive part of the E-step is the update for  $\phi$ . By running column-distributed EM, we are parallelizing most computation in updating  $\phi$ , the time complexity of updating  $\phi$  in each E-step hence decreases from  $O((Nk^2 + NMku)t_E)$  to  $O((Nk^2 + Nku)t_E)$ . In the M-step, the cost of updating  $\beta$  decreases from  $O(NMku)$  to  $O(Nku)$  through parallelization.

## 6. GENERALIZED BCE

Most cluster ensemble algorithms only combine the base clustering results to generate a consensus clustering, which might not be a good use of data when features of the original data points are available, and meanwhile, the performance of the ensemble algorithm is highly restricted by the base clustering algorithms, that is, the chances of obtaining a good consensus clustering from a set of very poor base clustering algorithms are low. In this section, we propose a GBCE algorithm which overcomes the two drawbacks by combining both the base clustering results and feature vectors of original data points to yield a consensus clustering.

The main idea of GBCE is as follows: For each data point  $i$ , we concatenate its  $D$ -dimensional feature vector  $\mathbf{o}_i$  after the  $M$ -dimensional base clustering vector  $\mathbf{x}_i$  to get an  $(M+D)$ -dimensional vector  $\mathbf{y}_i$ . Following Shan and Banerjee [15], the generative process for  $\mathbf{y}_i$  is given as follows:

1. Choose  $\theta_i \sim \text{Dirichlet}(\alpha)$ .
2. For each nonmissing base clustering, that is,  $y_{ij}, [j]_1^M$ :
  - (a) Choose a component  $z_{ij} = h \sim \text{discrete}(\theta_i)$ ;
  - (b) Choose the base clustering result  $y_{ij} \sim \text{discrete}(\beta_{hj})$ .
3. For each nonmissing feature of the data point, that is,  $y_{ij}, [j]_{M+1}^{M+D}$ :
  - (a) Choose a component  $z_{ij} = h \sim \text{discrete}(\theta_i)$ ;
  - (b) Choose the feature value  $y_{ij} \sim p_{\psi_j}(y_{ij}|\zeta_{hj})$ .

The first two steps are the same with BCE. The difference is the new step 3, where each feature of the data point is generated from  $p_{\psi_j}(y_{ij}|\zeta_{hj})$ —an exponential family

distribution for feature  $j$  and consensus cluster  $h$  [15].  $\psi_j$  in  $p_{\psi_j}(y_{ij}|\zeta_{hj})$  determines a particular family for feature  $j$ , such as Gaussian, Poisson, etc., and  $\zeta_{hj}$  determines a particular parameter for the distribution in that family. For the ease of exposition, we assume that all original features have real values generated from Gaussian distributions, then  $p_{\psi_j}(y_{ij}|\zeta_{hj})$  could be denoted by [15]

$$p(y_{ij}|\mu_{hj}, \sigma_{hj}^2) = \frac{1}{\sqrt{2\pi\sigma_{hj}^2}} \exp\left(-\frac{(y_{ij} - \mu_{hj})^2}{2\sigma_{hj}^2}\right),$$

where  $\mu_{hj}$  and  $\sigma_{hj}^2$  are the mean and variance for the Gaussian distribution of cluster  $h$  and feature  $j$ .

The proposed generative model gives an intuitive way to combine the feature vectors with base clustering results, but it suffers from a limitation that it treats each feature of the original data point as important as each base clustering result. In such cases, for high dimensional data points with  $D \gg M$ , the feature vectors will dominate the consensus clustering, i.e., the consensus clustering result is almost the same with running the algorithm on only the data points without base clustering results. Therefore, we further generalize the algorithm to allow different weights for different base clustering results and different features, yielding generalized BCE.

Given non-negative integral weights  $\mathbf{u} = \{u_j, [j]_1^{M+D}\}$  for  $y_{ij}, [j]_1^{M+D}$ , the generative process of GBCE for  $\mathbf{y}_i$  with weight  $\mathbf{u}$  is given as follows:

1. Choose  $\boldsymbol{\theta}_i \sim \text{Dirichlet}(\boldsymbol{\alpha})$ .
2. For each nonmissing base clustering, that is,  $y_{ij}, [j]_1^M$ , repeat 2(a) and 2(b) for  $u_j$  times:
  - (a) Choose a component  $z_{ij} = h \sim \text{discrete}(\boldsymbol{\theta}_i)$ ;
  - (b) Choose the base clustering result  $y_{ij} \sim \text{discrete}(\beta_{hj})$ .
3. For each nonmissing original feature, that is,  $y_{ij}, [j]_{M+1}^{M+D}$ , repeat 3(a) and 3(b) for  $u_j$  times:
  - (a) Choose a component  $z_{ij} = h \sim \text{discrete}(\boldsymbol{\theta}_i)$ ;
  - (b) Choose the feature value  $y_{ij} \sim N(\mu_{hj}, \sigma_{hj}^2)$ .

Therefore, BCE is a special case of GBCE by setting  $u_j = 1$  for  $[j]_1^M$  and  $u_j = 0$  for  $[j]_{M+1}^{M+D}$ . The marginal probability

for weighted  $\mathbf{y}_i$  is given by

$$\begin{aligned} p(\mathbf{y}_i|\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\mu}, \boldsymbol{\sigma}^2, \mathbf{u}) & \quad (11) \\ &= \int_{\boldsymbol{\theta}_i} p(\boldsymbol{\theta}_i|\boldsymbol{\alpha}) \prod_{j=1, \exists y_{ij}}^M \left( \sum_h p(z_{ij} = h|\boldsymbol{\theta}_i) p(y_{ij}|\beta_{hj}) \right)^{u_j} \\ & \quad \prod_{j=M+1, \exists y_{ij}}^{M+D} \left( \sum_h p(z_{ij} = h|\boldsymbol{\theta}_i) p(y_{ij}|\mu_{hj}, \sigma_{hj}^2) \right)^{u_j} d\boldsymbol{\theta}_i. \end{aligned}$$

In GBCE, if we set  $u_j = 1$  for  $[j]_{M+1}^{M+D}$  and  $u_j = D$  for  $[j]_1^M$ , we are treating each base clustering as important as the whole feature vector of the data point, instead of a single feature. We can also set different weights for different  $y_j$  based on the confidence of clustering accuracy, or importance of the feature, etc. In addition, in the generative process, the weights have been assumed to be non-negative integers since they denote the repetition times, but the learning algorithm we discuss below still holds even when  $u_j$  is generalized to positive real numbers, yielding a very flexible model.

From the generative process, GBCE actually does not generate  $\mathbf{y}_i$ , but generates a new vector  $\tilde{\mathbf{y}}_i$  with  $y_{ij}$  repeated for  $u_j$  times to incorporate the weights. However, we do not need to create a  $\tilde{\mathbf{y}}_i$  explicitly to learn the model. For inference and parameter estimation, similar with Section 5, we introduce a family of variational distributions

$$q(\boldsymbol{\theta}_i, \mathbf{z}_i|\gamma_i, \phi_i, \mathbf{u}) = q(\boldsymbol{\theta}_i|\gamma_i) \prod_{j=1}^{M+D} q(z_{ij}|\phi_{ij})^{u_j}$$

to approximate  $p(\boldsymbol{\theta}_i, \mathbf{z}_i|\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{u}, \mathbf{y}_i)$ . The update equations for variational parameters are given by

$$\phi_{ijh} \propto \exp\left(\Psi(\gamma_{ih}) - \Psi\left(\sum_{h'} \gamma_{ih'}\right)\right) \quad (12)$$

$$+ \sum_{r=1}^{k_j} \mathbb{1}(r|i, j) \log \beta_{hj}(r), \quad [j]_1^M,$$

$$\phi_{ijh} \propto \exp\left(\Psi(\gamma_{ih}) - \Psi\left(\sum_{h'} \gamma_{ih'}\right)\right) \quad (13)$$

$$- \log \sigma_{hj} - \frac{(y_{ij} - \mu_{hj})^2}{2\sigma_{hj}^2}, \quad [j]_{M+1}^{M+D},$$

$$\gamma_{ih} = \alpha_h + \sum_{j=1, \exists y_{ij}}^{M+D} u_j \phi_{ijh}, \quad (14)$$

where  $[i]_1^N$  and  $[h]_1^k$ . For the model parameters, the update equations for  $\boldsymbol{\alpha}$  is the same as in Eq. (8), and the equations

for the rest of parameters are given by

$$\beta_{hj}(r) \propto u_j \sum_{i=1}^N \phi_{ijh} \mathbb{1}(r|i, j), \quad (15)$$

$$\mu_{hj} = \frac{\sum_{i=1, \exists x_{ij}}^N u_j \phi_{ijh} y_{ij}}{\sum_{i=1, \exists x_{ij}}^N u_j \phi_{ijh}}, \quad (16)$$

$$\sigma_{hj}^2 = \frac{\sum_{i=1, \exists x_{ij}}^N u_j \phi_{ijh} (y_{ij} - \mu_{hj})^2}{\sum_{i=1, \exists x_{ij}}^N u_j \phi_{ijh}}, \quad (17)$$

where  $[h]_1^k$ ,  $[j]_1^M$ , and  $[r]_1^k$ .

## 7. EXPERIMENTAL RESULTS

In this section, we run experiments on datasets from UCI machine learning repository and KDD Cup 1999. In particular, for UCI data, we pick 12 datasets which are relatively small. (For wine quality we only keep the data points in three main classes, so the classes with very few number of data points are removed.) For KDD Cup data, there are four main classes among 37 classes in total. We randomly pick 2 000 000 data points from these four main classes and divide them into two parts, so we have two relatively large datasets with one million data points each. The number of objects, features and classes in each data set are listed in Table 1, where kdd99-1 and kdd99-2 are from KDD Cup 1999 and the rest are from UCI machine learning repository.

For all reported results, there are two steps leading to the final consensus clustering. First, we run base clustering algorithms to get a set of base clustering results. Second, various cluster ensemble algorithms, including MM [1],

**Table 1.** The number of the instances, features, and classes in each dataset.

Dataset	Instances	Features	Classes
pima	768	8	2
iris	150	4	3
wdbc	569	30	2
balance	625	4	3
glass	214	9	6
bupa	345	6	2
wine	178	13	3
magic04	19020	10	2
ionosphere	351	34	2
segmentation	2100	19	7
kdd99-1	1 000 000	41	4
kdd99-2	1 000 000	41	4
chess	3196	36	2
wine quality	4535	11	3

CSPA, HGPA, MCLA [2] and  $k$ -means, are applied to the base clustering results to generate a consensus clustering. We compare their results with BCE.

The comparison between BCE and other cluster ensemble algorithms are divided into five categories as follows:

1. General cluster ensemble (general).
2. Cluster ensemble with missing values (miss-v).
3. Cluster ensemble with increasing number of columns (increase-c), that is, additional base clusterings.
4. Column-distributed cluster ensemble (column-d).
5. Row-distributed cluster ensemble (row-d).

Table 2 shows the five categories of experiments and the six cluster ensemble algorithms we use. We can see that most of the algorithms can only accomplish a few tasks among the five. In principle, MM can be generalized to deal with all five scenarios; however, the literature does not have an explicit algorithm for column- or row-distributed cluster ensembles using MM. As we can see from Table 2, BCE is the most flexible and versatile among the six algorithms.

For evaluation, we use micro-precision [18] to measure accuracy of the consensus cluster with respect to the true labels: the micro-precision is defined as

$$MP = \sum_{h=1}^k a_h / n,$$

where  $k$  is the number of clusters and  $n$  is the number of objects,  $a_h$  denotes the number of objects in consensus cluster  $h$  that are correctly assigned to the corresponding class. We identify the ‘corresponding class’ for consensus cluster  $h$  as the true class with the largest overlap with the cluster, and assign all objects in cluster  $h$  to that class. Note that  $0 \leq MP \leq 1$  with 1 indicating the best possible consensus clustering, which has to be in full agreement with the class labels.

**Table 2.** The applicability of algorithms to different experimental settings:  $\checkmark$  indicates that the algorithm is applicable, and  $\times$  indicates otherwise.

Algorithm	General	Miss-v	Increase-c	Column-d	Row-d
$k$ -means	$\checkmark$	$\times$	$\checkmark$	$\times$	$\checkmark$
CSPA	$\checkmark$	$\checkmark$	$\checkmark$	$\times$	$\times$
HGPA	$\checkmark$	$\checkmark$	$\checkmark$	$\times$	$\times$
MCLA	$\checkmark$	$\checkmark$	$\checkmark$	$\times$	$\times$
MM	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$
BCE	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$

**Table 3.**  $k$ -means with different initializations are used as base clustering algorithms: (a) Maximum  $MP$  for different cluster ensemble algorithms; (b) average  $MP$  for different cluster ensemble algorithms (Magic04, kdd99-1, and kdd99-2 are too large so CSPA could not finish its run). The highest  $MP$  among different algorithms on each data set is bolded.

	Base clusterings		Cluster ensembles				
	$k$ -means	MCLA	CSPA	HGPA	MM	$k$ -means	BCE
<i>(a) Maximum MP</i>							
iris	0.8867	0.8867	0.9533	0.7333	0.9067	0.5267	<b>0.9600</b>
wdbc	0.8541	0.8840	0.8840	0.5518	0.8840	0.8840	<b>0.8893</b>
ionosphere	0.7123	0.7123	0.6952	0.6353	0.7179	0.7094	<b>0.7749</b>
glass	0.5421	0.5187	0.4393	0.4439	0.5748	0.5093	<b>0.6121</b>
bupa	0.4841	0.5652	0.5710	0.5188	0.5710	0.5565	<b>0.5942</b>
pima	0.6602	0.6602	0.5065	0.5260	0.6654	0.6029	<b>0.7044</b>
wine	0.6629	0.7247	<b>0.7416</b>	0.5562	0.7247	0.4775	0.7247
magic04	0.6491	0.6491	×	0.6491	0.6530	0.6491	<b>0.6531</b>
balance	0.5936	0.5216	0.5408	0.4256	<b>0.6016</b>	0.5824	0.5968
segmentation	0.5710	0.5657	0.5810	0.5419	0.6233	0.5710	<b>0.6362</b>
kdd99-1	0.7458	0.7703	×	0.6846	0.7652	0.7227	<b>0.7827</b>
kdd99-2	0.7642	0.7475	×	0.7352	0.7651	0.7523	<b>0.7804</b>
<i>(b) Average MP</i>							
iris	0.6267	0.8867	<b>0.9167</b>	0.7333	0.8867	0.5267	0.8911
wdbc	0.7595	<b>0.8840</b>	<b>0.8840</b>	0.5188	<b>0.8840</b>	0.8689	<b>0.8840</b>
ionosphere	0.6906	0.7046	0.6952	0.6063	0.7111	0.7094	<b>0.7123</b>
glass	0.5140	0.4766	0.4393	0.4234	0.5519	0.4363	<b>0.5526</b>
bupa	0.4537	0.5652	<b>0.5710</b>	0.5075	0.5586	0.5164	0.5664
pima	0.5751	0.6602	0.5065	0.5163	0.6503	0.6029	<b>0.6612</b>
wine	0.5904	0.7247	<b>0.7416</b>	0.5250	0.7129	0.4775	0.7247
magic04	0.6252	0.6491	×	0.6235	0.6231	0.6250	<b>0.6497</b>
balance	0.5114	0.5188	0.5408	0.4256	<b>0.5514</b>	0.5824	0.5293
segmentation	0.5574	0.5657	0.5810	0.4543	0.5817	0.5142	<b>0.5854</b>
kdd99-1	0.6281	<b>0.7689</b>	×	0.6621	0.7411	0.5899	0.7642
kdd99-2	0.6643	0.7475	×	0.6720	0.7463	0.7015	<b>0.7523</b>

In the following subsections, we will present the experimental results for five categories of experiments as in Table 2, starting from general cluster ensembles.

### 7.1. General Cluster Ensembles

In this subsection, we run two types of experiments: one only uses  $k$ -means as the base clustering algorithms, and the other uses multiple algorithms as the base clustering algorithms.

Given  $N$  objects, we first use  $k$ -means as the base clustering algorithm on 12 datasets. For ten UCI datasets, we run  $k$ -means 2000 times with different initializations to obtain 2000 base clustering results, which are divided evenly into 100 subsets, with 20 base clustering results in each of them. For two large datasets kdd99-1 and kdd99-2, we run the experiments following the same strategy, but we keep three subsets with five base clustering results in each of them. Cluster ensemble algorithms are then applied on each subset. The maximum and average  $MP$ s over all subsets are reported in Tables 3(a) and 3(b).

We also use  $k$ -means, fuzzy  $c$ -means (FCM) [19], METIS [8], and affinity propagation (AP) [20] as the base

clustering algorithms on 11 datasets for cluster ensemble.<sup>1</sup> By running  $k$ -means 500 times, FCM 800 times, METIS 200 times, and AP 500 times with different initializations, we also obtain 2000 base clustering results. Following the same strategy above to run cluster ensemble algorithms, we have the maximum and average  $MP$ s in Table 5(a) and 5(b).

The key observations from Table 3 and 5 can be summarized as follows: (i) BCE almost always has a higher max and average  $MP$  than base clustering results, which means the consensus clustering from BCE is indeed better in quality than the original base clusterings. (ii) BCE outperforms other cluster ensemble algorithms for most of the times in terms of both maximum and average  $MP$ , no matter which base clustering algorithms are used.

Since the results of MM and BCE are rather close to each other, to make a careful comparison, we run a paired  $t$ -test under the hypothesis

$$H_0 : MP(\text{MM}) = MP(\text{BCE}),$$

<sup>1</sup> We run this set of experiments on relatively small datasets since METIS and AP cannot run on large ones such as kdd99-1 and kdd99-2.

**Table 4.**  $k$ -means with different initializations are used as base clustering algorithms: Paired  $t$ -test for MM and BCE, where ‘Mean-D’ is the mean of  $MP$  differences obtained by (MM-BCE), and ‘sd-MM (BCE)’ is standard deviation of the  $MP$ s from MM (BCE). For ‘Mean-D’, the datasets where BCE performs better is bolded. For ‘sd-MM’ and ‘sd-BCE’, the one with a smaller standard deviation is bolded.

Dataset	Mean-D	sd-MM	sd-BCE	$p$ -value
iris	<b>-0.0402</b>	0.0221	<b>0.0103</b>	0.0026
wdbc	<b>-0.0009</b>	0.0018	<b>0.0000</b>	0.3256
ionosphere	<b>-0.0013</b>	0.0024	<b>0.0000</b>	0.0169
glass	<b>-0.0046</b>	0.0110	<b>0.0076</b>	0.0511
bupa	<b>-0.0128</b>	0.0377	<b>0.0013</b>	0.0018
pima	<b>-0.0117</b>	0.0205	<b>0.0038</b>	0.0089
wine	<b>-0.0240</b>	0.0290	<b>0.0119</b>	0.0239
magic04	<b>-0.0010</b>	0.0023	<b>0.0014</b>	0.4127
balance	0.0301	0.0384	<b>0.0061</b>	0.9990
segmentation	<b>-0.0140</b>	0.0331	<b>0.0186</b>	0.2250
kdd99-1	<b>-0.0120</b>	0.0207	<b>0.0091</b>	0.0382
kdd99-2	<b>-0.0082</b>	0.0103	<b>0.0031</b>	0.0534

**Table 6.**  $k$ -means, FCM, AP, and METIS are used as the base clustering algorithms: Paired  $t$ -test for MM and BCE, where ‘Mean-D’ is the mean of  $MP$  differences obtained by (MM-BCE), and ‘sd-MM (BCE)’ is standard deviation of the  $MP$ s from MM (BCE). For ‘Mean-D’, the datasets where BCE performs better is bolded. For ‘sd-MM’ and ‘sd-BCE’, the one with a smaller standard deviation is bolded.

Dataset	Mean-D	sd-MM	sd-BCE	$p$ -value
iris	<b>-0.0104</b>	<b>0.0182</b>	0.0204	0.0047
wdbc	<b>-0.0007</b>	0.0022	<b>0.0000</b>	0.2813
ionosphere	<b>-0.0058</b>	0.0061	<b>0.0002</b>	0.0081
glass	0.0072	<b>0.0030</b>	0.0062	0.9681
bupa	0.0110	0.0224	<b>0.0021</b>	0.9961
pima	<b>-0.0120</b>	0.0216	<b>0.0008</b>	0.0046
wine	<b>-0.0121</b>	0.0204	<b>0.0082</b>	0.0061
balance	0.0248	0.0428	<b>0.0042</b>	0.9939
segmentation	<b>-0.0040</b>	0.0271	<b>0.0092</b>	0.3145
chess	<b>-0.0140</b>	0.0094	<b>0.0086</b>	0.0304
wine quality	<b>-0.0172</b>	<b>0.0006</b>	0.0024	0.0112

$$H_a : MP(\text{MM}) < MP(\text{BCE}).$$

The test is designed to assess the strength of the evidence against  $H_0$  and supporting  $H_a$ . Such strength is measured by the  $p$ -value, a lower  $p$ -value indicates stronger evidence.

In our case, a lower  $p$ -value indicates that the performance improvements of BCE over MM is statistically significant. Usually a  $p$ -value less than 0.05 is considered as strong evidence. The results are shown in Tables 4 and 6, respectively. BCE outperforms MM with a low  $p$ -value ( $<0.05$ ) most of the times, indicating that  $MP(\text{BCE})$  is

**Table 5.**  $k$ -means, FCM, AP, and METIS are used as the base clustering algorithms: (a) Maximum  $MP$  for different cluster ensemble algorithms; (b) average  $MP$  for different cluster ensemble algorithms. The highest  $MP$  among different algorithms on each data set is bolded.

	Base clusterings			Cluster ensembles					
	$k$ -means	FCM	AP	METIS	MCLA	CSPA	HGPA	MM	BCE
<i>(a) Maximum MP</i>									
iris	0.8867	0.8933	0.8867	0.8867	0.8893	0.9533	0.7431	0.9067	<b>0.9600</b>
wdbc	0.8541	0.8541	0.8541	0.8541	0.8840	0.8840	0.6731	0.8840	<b>0.8893</b>
ionosphere	0.7123	0.7123	0.7094	0.6806	0.7046	0.6952	0.6782	0.7179	<b>0.7655</b>
glass	0.5421	0.5270	0.5224	0.5317	0.5187	0.4566	0.4439	0.5822	<b>0.6052</b>
bupa	0.4841	0.4548	0.4537	0.4541	0.5524	0.5710	0.5234	0.5710	<b>0.5855</b>
pima	0.6602	0.6602	0.6602	0.6602	0.6602	0.5125	0.5260	0.6654	<b>0.7044</b>
wine	0.6629	0.6629	0.6068	0.6571	0.7247	<b>0.7416</b>	0.5428	0.7247	0.7247
balance	0.5936	0.5612	0.5091	0.5621	0.5112	0.5408	0.4452	<b>0.6016</b>	0.5848
segmentation	0.5710	0.4356	0.5710	0.4524	0.5657	0.5810	0.5419	0.6233	<b>0.6362</b>
chess	0.4000	0.5466	0.5466	0.4408	0.5466	0.5025	0.5003	0.5466	<b>0.5529</b>
wine quality	0.4563	0.4008	0.3800	0.3563	0.4646	0.3857	0.3563	0.4646	<b>0.5078</b>
<i>(b) Average MP</i>									
iris	0.6267	0.7468	0.7731	0.8064	0.8893	<b>0.9133</b>	0.7431	0.8893	0.8933
wdbc	0.7595	0.8200	0.7932	0.8175	<b>0.8840</b>	<b>0.8840</b>	0.5426	<b>0.8840</b>	<b>0.8840</b>
ionosphere	0.6906	0.6906	0.6721	0.6538	0.7046	0.6952	0.6104	0.7088	<b>0.7141</b>
glass	0.5140	0.4808	0.4981	0.5222	0.4810	0.4566	0.4314	<b>0.5506</b>	0.5435
bupa	0.4537	0.4461	0.4431	0.4260	0.5524	<b>0.5710</b>	0.4862	0.5586	0.5478
pima	0.5751	0.5541	0.5211	0.5460	0.6602	0.5125	0.5163	0.6496	<b>0.6621</b>
wine	0.5904	0.6062	0.5704	0.5521	0.7247	<b>0.7416</b>	0.5182	0.7129	0.7247
balance	0.5114	0.4804	0.4966	0.5072	0.5091	0.5408	0.4451	<b>0.5552</b>	0.5301
segmentation	0.5574	0.4238	0.5437	0.4097	0.5657	0.5810	0.4543	0.5817	<b>0.5854</b>
chess	0.3806	0.4560	0.4108	0.4281	0.5466	0.5025	0.4803	0.5321	<b>0.5500</b>
wine quality	0.2908	0.3602	0.3405	0.3563	0.4451	0.3857	0.3563	0.4646	<b>0.4802</b>

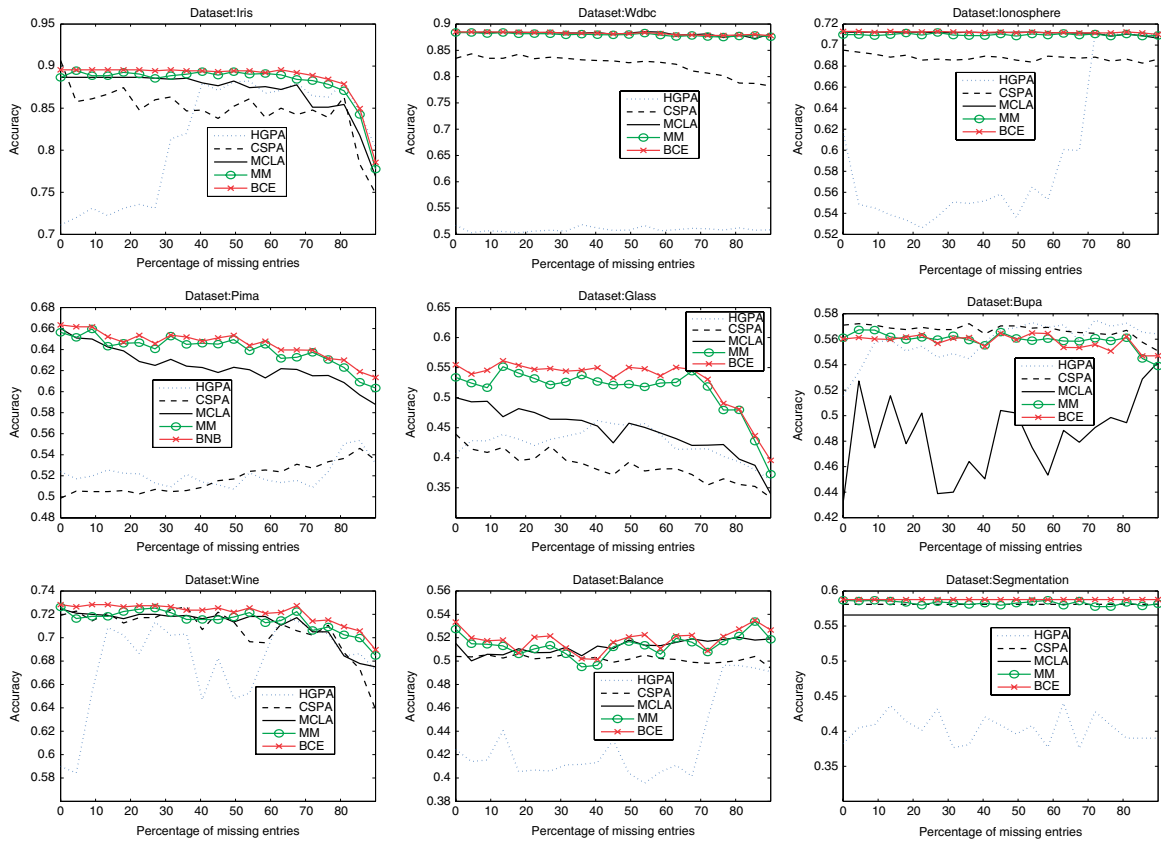


Fig. 3 Average  $MP$  with increasing percentage of missing values.

significantly better than  $MP(MM)$  on these datasets. In addition, the smaller standard deviation of BCE shows that it is more stable than MM.

## 7.2. Cluster Ensembles with Missing Values

Given 20 base clustering results for  $N$  objects, we randomly hold out  $p$  percent of data as missing values, with  $p$  increasing from 0 to 90 in steps of 4.5.<sup>2</sup> We compare the performance of different algorithms except  $k$ -means, because  $k$ -means cannot handle missing values. Each time we run the algorithms ten times and report  $MP$  on nine datasets in Fig. 3. Surprisingly, before the missing value percentage reaches 70%, most algorithms have a stable  $MP$  with increasing number of missing entries, without a distinct decrease in accuracy. BCE is always among the top one or two in terms of the accuracy across different percentage of missing values, indicating that BCE is one of the best algorithms to deal with missing value cluster ensemble. Comparatively, HGPA seems to have the worst performance in terms of both the accuracy and stability.

<sup>2</sup> Starting from this subsection, we only use  $k$ -means as the base clustering algorithm.

## 7.3. Cluster Ensembles with Increasing Columns

In order to find out the effect on the cluster ensemble accuracy with increasing number of base clusterings, we perform experiments for cluster ensemble with columns (base clusterings) increasing from 1 to 20 in steps of 1. We first generate 20 base clusterings as a pool. At each step  $s$ , we randomly pick  $s$  base clusterings from the pool, which is repeated for 50 times to generate 50 ( $N \times s$ ) base clustering matrices (note there are repetitions among these 50 matrices). We then run cluster ensemble on each of them. The average of  $MP$  over 50 runs at each step is reported in Fig. 4 for nine datasets.

First, we can see that BCE is again among the top one or two on all the data sets in our experiments. Second,  $MP$ s for most of the algorithms increase dramatically when the number of base clusterings increases from 1 to 5. After that, no distinct increase is observed. On Pima, the accuracy even decreases when the number of base clustering is larger than 10, which is possibly due to the poor performance of the base clusterings. The trends of the curves might be related to the diversity of the base clusterings. In our experiments, we only use  $k$ -means for all base clusterings, so the cluster information may become redundant after a certain number

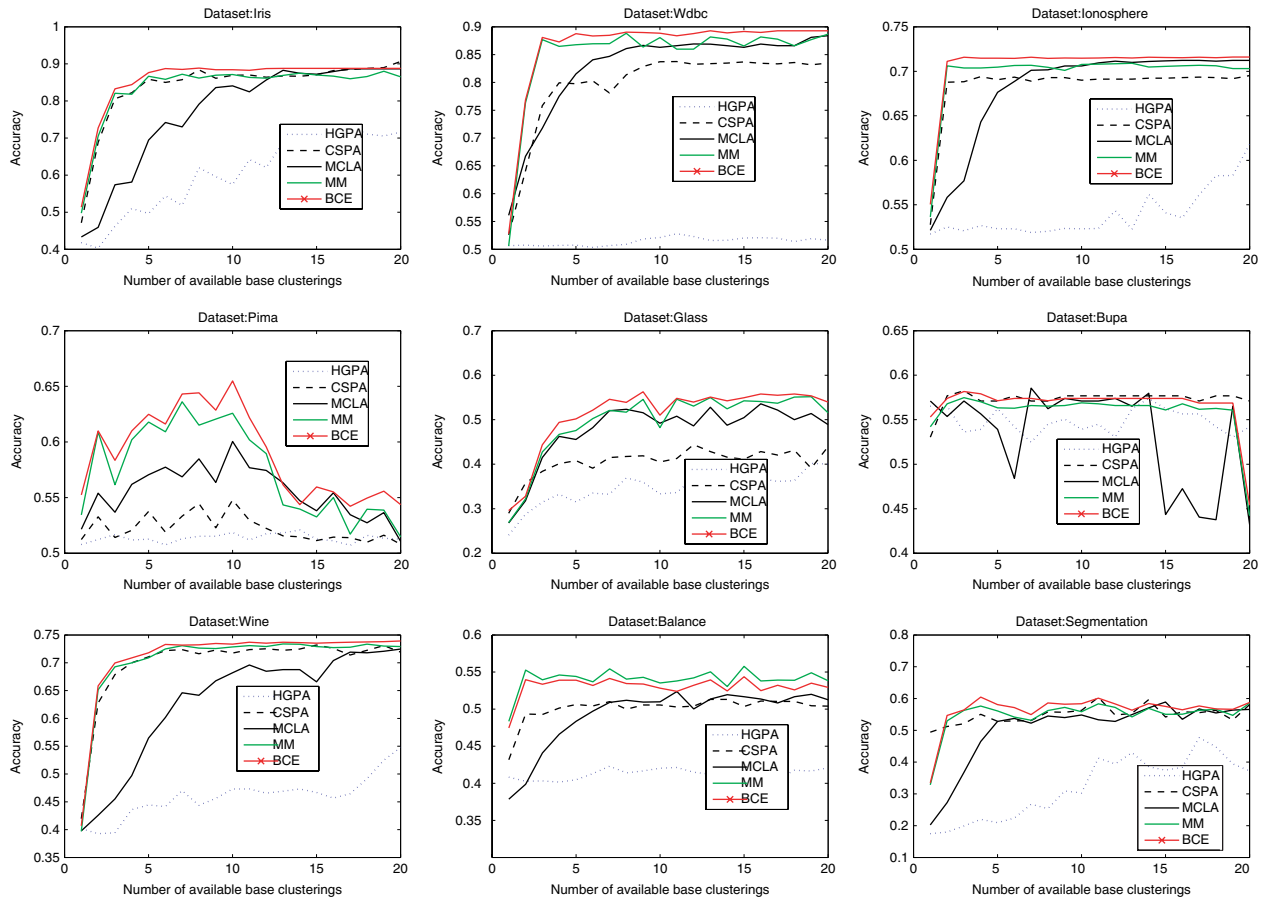


Fig. 4 Average  $MP$  comparison with increasing number of available base clusterings.

of base clusterings have been used, and the accuracy does not increase anymore. The accuracy may keep on increasing with more columns if the base clusterings are generated by different algorithms.

#### 7.4. Row-Distributed Cluster Ensembles

For experiments on row-distributed cluster ensembles, we divide our 20 base clustering results by rows (approximately) evenly into  $P$  partitions, with  $P$  increasing from 1 to 10 in steps of 1. We compare the performance of row-distributed BCE with distributed  $k$ -means [4]. Note that in our experiments, we use the heuristic row-distributed EM as in Section 5.2.1. Although no theoretical guarantee for convergence is provided, in our observation, the algorithm stops when model parameters do not change anymore within ten iterations. The comparative results on nine datasets are presented in Fig. 5. It is clear that row-distributed BCE always has a higher accuracy than distributed  $k$ -means except on Balance. For most datasets, the performance of row-distributed BCE is more stable across varying number of partitions, indicating its robustness.

#### 7.5. Column-Distributed Cluster Ensembles

We run experiments for column-distributed cluster ensembles with increasing number of base clusterings (20, 60, 120, 240, 480, 960, 1440, 1920), which are picked randomly from a pool of 3000 base clustering results. We run the client-server style algorithm as in Section 5.2.2 with one client maintaining one base clustering, such that multiple clients could run in parallel. The accuracy in the column-distributed case would be the same as the general cluster ensemble using BCE since they are using exactly the same algorithm except that the column-distributed variants run it in a distributed manner. If we ignore the communication overhead between the clients and server, the comparison of running time between the column-distributed and general cluster ensemble is presented in Fig. 6. We can see that column-distributed cluster ensemble is much more efficient than the general case, especially when the number of base clusterings is large, the column-distributed variant is several orders of magnitudes faster. Therefore, the column-distributed BCE is readily applicable to the real-life settings with large data sets.

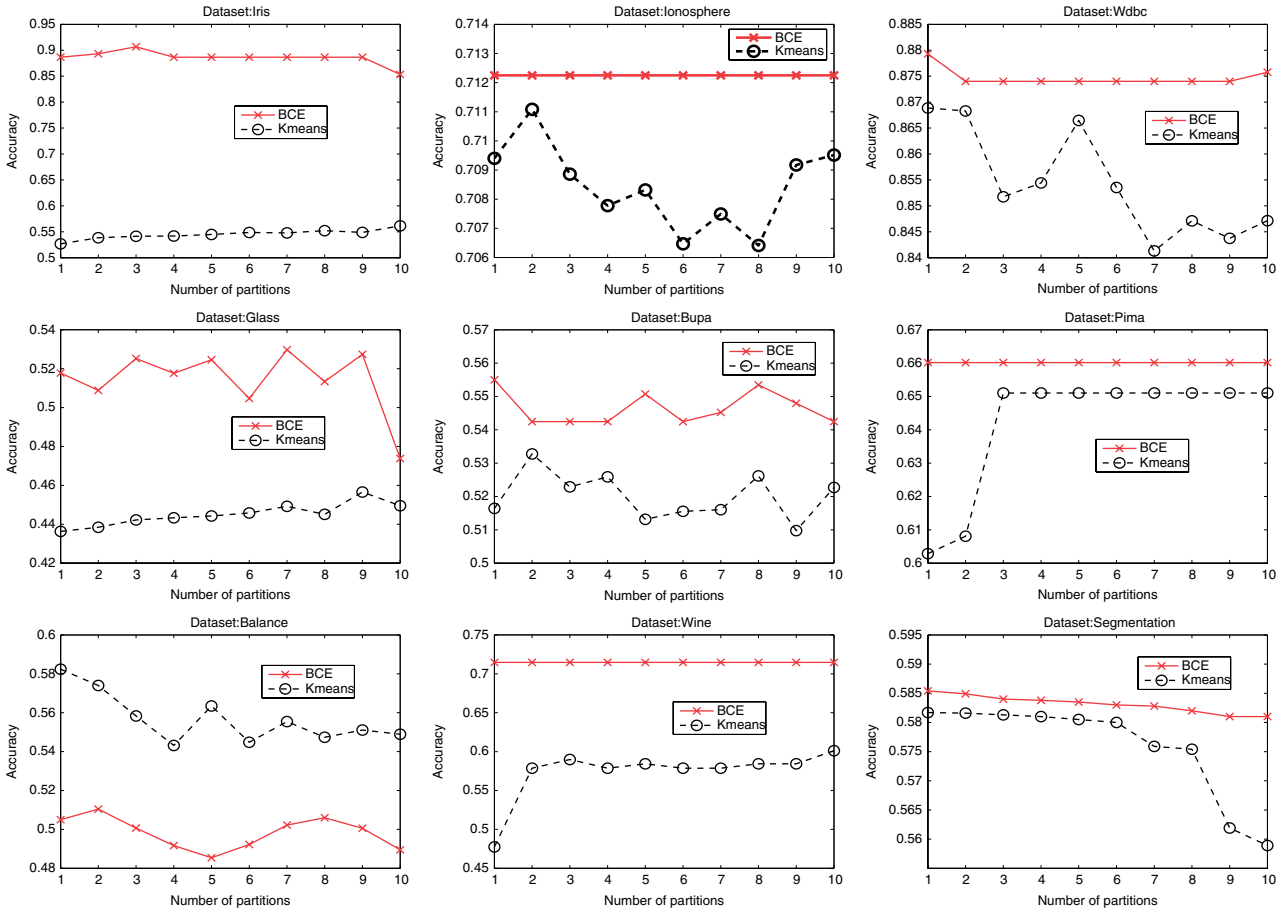


Fig. 5 Average  $MP$  with increasing number of distributed partitions.

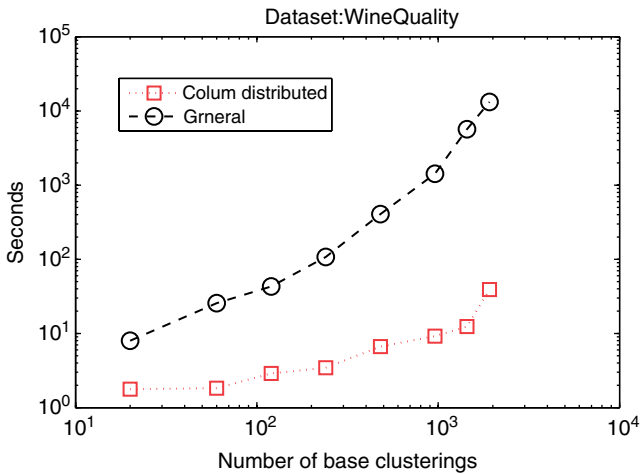


Fig. 6 The comparison of running time between column-distributed and general cluster ensemble.

### 7.6. Generalized Cluster Ensembles

To compare GBCE to BCE, we set the weight of each original feature to be 1 and the weight of each base

clustering result to be  $D$  in GBCE, where  $D$  is the number of features in original data points. Similar with Section 7.3, we first generate 50 base clustering results as a pool, then at each step  $s$ , we randomly pick  $s$  base clusterings from the pool to run GBCE and BCE, where  $s$  increases from 1 to 10. Fig. 7 show the final result averaged over 50 runs at each step  $s$ .<sup>3</sup> We do not show results on bupa since the accuracy on bupa from GBCE and BCE are exactly the same.

Overall, Fig. 7 contains two cases of the comparison: In the first case (on wdbc, ionosphere, pima, magic04, and balance), the whole curve of GBCE is (mostly) above BCE, which shows a clear improvement by combining original data points with base clustering results. In the second case (on iris, glass, wine, segmentation), GBCE has higher accuracy when there is only a small number of base clustering results, and there is no clear winner when the number of base clustering results increases. It is probably because the base clustering algorithms generate

<sup>3</sup> The base clusterings we use are different from Section 7.3, so the result of BCE is different from Fig. 4.

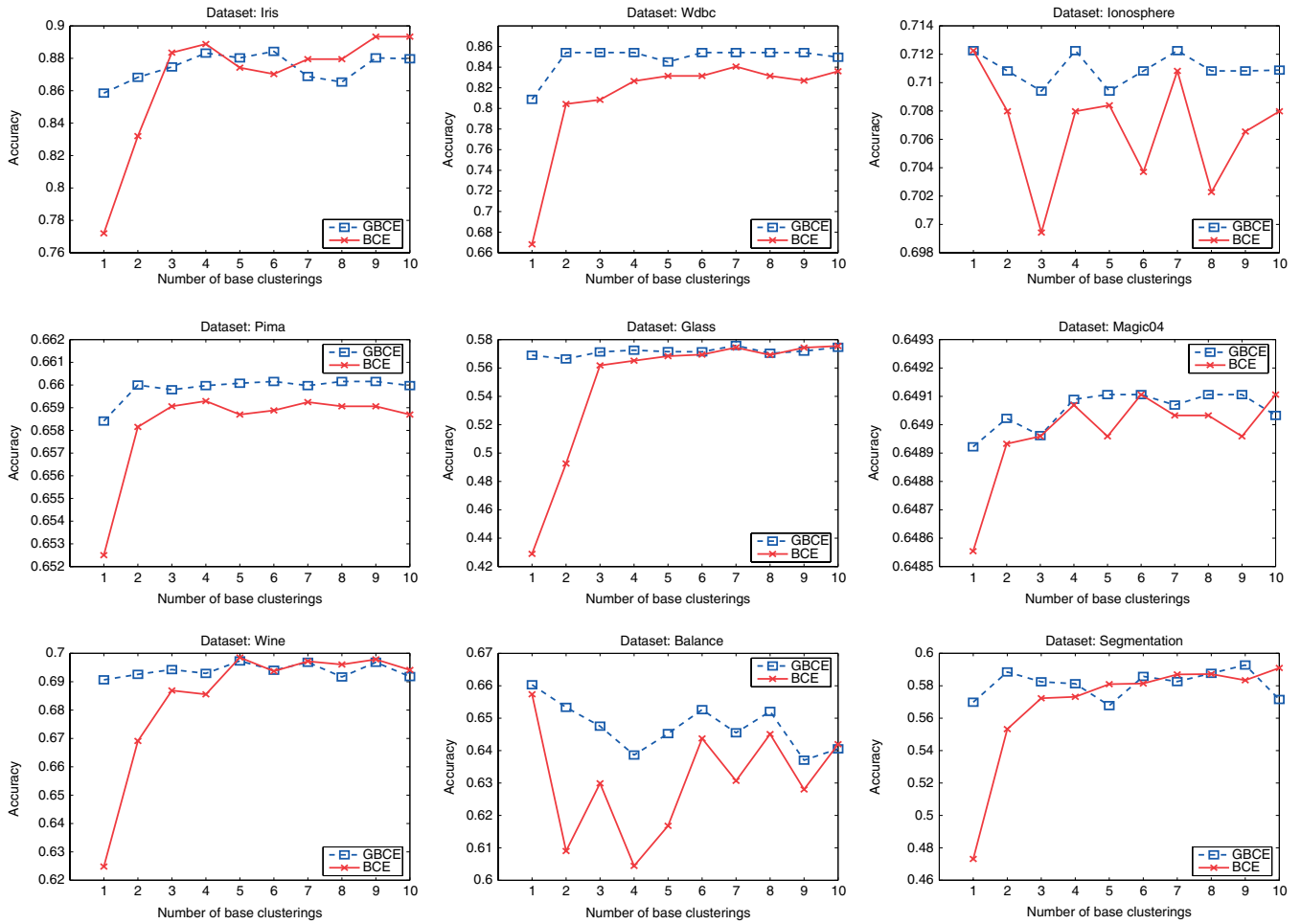


Fig. 7 Average *MP* comparison between BCE and GBCE with increasing number of available base clusterings.

good clusterings, so when more base clustering results are combined together, BCE performs as good as GBCE. In addition, we also run experiments for GBCE with different weights for the base clustering results varying from  $D/8$  to  $8D$  in multiplicative steps of 2. There is no clear trend with increasing weights. Generally, the results with the weight larger than  $D$  are quite similar with each other, and the results with the weight smaller than  $D$  could be different, but the difference decreases when the number of base clustering results increases. We show two examples with weights  $\{D/8, D/4, D/2, D\}$  in Fig. 8.

## 8. CONCLUSION

In this paper, we have proposed Bayesian cluster ensembles, a mixed-membership generative model for obtaining a consensus clustering by combining multiple base clustering results. BCE provides a Bayesian way

to combine clusterings, and entirely avoids cluster label correspondence problems encountered in graph based approaches to the cluster ensemble problem. A variational approximation based algorithm is proposed for learning a Bayesian cluster ensemble. In addition, we have also proposed GBCE, which generates a consensus clustering by taking both the base clustering results and original data points. Compared with existing algorithms, BCE is the most versatile because of its applicability to several variants of the cluster ensemble problem, including missing value cluster ensembles, row-distributed and column-distributed cluster ensembles. In addition, extensive experimental results show that BCE outperforms other algorithms in terms of accuracy and stability, and it can be run in a distributed manner without exchanging base clustering results, thereby preserving privacy and/or substantial speed-ups. Finally, the comparison between GBCE and BCE show that GBCE can generate higher accuracy than BCE, especially with only a small number of base clustering results available.

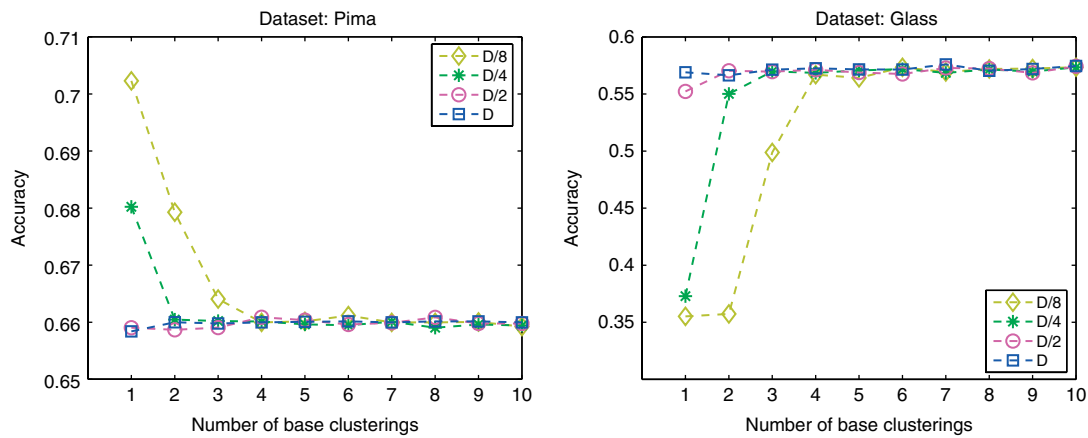


Fig. 8 GBCE with different weights on base clustering results.

### ACKNOWLEDGMENTS

This research was supported by NSF grants IIS-0812183, IIS-0916750, CNS-1017647, IIS-1029711, NSF CAREER grant IIS-0953274, NASA grant NNX08AC36A, and NSF grant 61003142.

### REFERENCES

- [1] A. Topchy, A. Jain, and W. Punch, A mixture model for clustering ensembles, In *SDM*, 2004, 379–390.
- [2] A. Strehl and J. Ghosh, Cluster ensembles—a knowledge reuse framework for combining multiple partitions, *JMLR* 3 (2002), 583–617.
- [3] L. Kuncheva and D. Vetrov, Evaluation of stability of k-means cluster ensembles with respect to random initialization, *PAMI* 28 (2006), 1798–1808.
- [4] G. Jagannathan and R. Wright, Privacy-preserving distributed k-means clustering over arbitrarily partitioned data, In *KDD*, 2005, 593–599.
- [5] X. Fern and C. Brodley, Solving cluster ensemble problems by bipartite graph partitioning, In *ICML*, 2004, 281–288.
- [6] M. Al-Razgan and C. Domeniconi, Weighted cluster ensemble, In *SDM*, 2006, 258–269.
- [7] X. Fern and C. Brodley, Random projection for high dimensional data clustering: a cluster ensemble approach, In *ICML*, 2003, 186–193.
- [8] G. Karypis and V. Kumar, A fast and high quality multilevel scheme for partitioning irregular graphs, *SIAM J Sci Comput* 20 (1999), 359–392.
- [9] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar, Multilevel hypergraph partitioning: applications in VLSI design, In *ACM/IEEE Design Automation Conference*, 1997, 526–529.
- [10] A. Fred and A. Jain, Data clustering using evidence accumulation, In *ICPR*, 2002, 276–280.
- [11] T. Li, C. Ding, and M. Jordan, Solving consensus and semi-supervised clustering problems using nonnegative matrix factorization, In *ICDM*, 2007.
- [12] P. Kellam, X. Liu, N. Martin, C. Orengo, S. Swift, and A. Tucker, Comparing, contrasting and combining clusters in viral gene expression data, In *Workshop on Intelligent Data Analysis in Medicine and Pharmacology*, 2001, 56–62.
- [13] S. Monti, P. Tamayo, J. Mesirov, and T. Golub, Consensus clustering: a resampling-based method for class discovery and visualization of gene expression microarray data, *Mach Learn J* 52 (2003), 91–118.
- [14] A. Banerjee and H. Shan, Latent Dirichlet conditional naive-Bayes models, In *ICDM*, 2007, 421–426.
- [15] H. Shan and A. Banerjee, Mixed-membership naive Bayes models. *Data Mining and Knowledge Discovery*, 2010.
- [16] D. Blei, A. Ng, and M. Jordan, Latent Dirichlet allocation, *JMLR* 3 (2003), 993–1022.
- [17] R. Neal and G. Hinton, A view of the EM algorithm that justifies incremental, sparse, and other variants, In *Learning in Graphical Models*, 1998, 355–368.
- [18] Z. Zhou and W. Tang, Clusterer ensemble, *Knowl Based Syst* 1 (2006), 77–83.
- [19] J. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum Press, New York, 1981.
- [20] B. Frey and D. Dueck, Clustering by passing messages between data points, *Science* 315 (2007), 972–976.