

# Bayesian Cluster Ensembles

Hongjun Wang\*

Hanhuai Shan †

Arindam Banerjee‡

## 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, as well as row-distributed 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 Ensembles (BCE), which is a mixed-membership model for learning cluster ensembles, and is applicable to all the primary variants of the problem. We propose two methods, respectively based on variational approximation and Gibbs sampling, for learning a Bayesian cluster ensemble. 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 the other algorithms in terms of stability and accuracy.

## 1 Introduction

Cluster ensembles provide a framework for combining multiple base clusterings of a dataset into a single consolidated clustering without accessing the features of the data or base clustering algorithms. Compared to individual clustering algorithms, cluster ensembles generate more robust and stable clustering results [21]. In principle, cluster ensembles can leverage distributed computing by calculating the base clusterings in an entirely distributed manner [20]. In addition, since cluster ensembles only need access to the base clustering results instead of the data itself, they provide a convenient approach to privacy preservation and knowledge reuse [20]. 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 applica-

tions. First, there may be missing values in the base clusterings. For example, in a customer segmentation application, while there will be legacy clusterings on old customers, there will be no result 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 considered as private information so that they cannot be shared. Cluster ensemble algorithms should be able to work with such “column-distributed” base clusterings. Third, the data objects 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 cluster ensemble algorithms handle such “row-distributed” base clusterings.

Current cluster ensemble algorithms, such as the cluster-based similarity partitioning algorithm (CSPA) [20], hypergraph partitioning algorithm (HGPA) [20], or  $k$ -means based algorithms [14] are able 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 [21] is applicable to the variants, although 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 approach using a Bayesian approach, i.e., by effectively maintaining a distribution over all possible consensus clusterings. It also seamlessly generalizes to all the important variants discussed above. Similar to the mix-

\*School of Computer Science, Sichuan University, Chengdu, China

†Dept of Computer Science & Engineering, University of Minnesota, Twin Cities

‡Dept of Computer Science & Engineering, University of Minnesota, Twin Cities

ture modeling approach, BCE treats all base clustering results for each object as a feature vector with discrete feature values, and learns a mixed-membership model from such a feature representation. Extensive empirical evaluation demonstrates that BCE is not only versatile in terms of its applicability, it mostly outperforms the other cluster ensemble algorithms in terms of stability and accuracy.

The rest of the paper is organized as follows. In Section 2, we give a problem definition. In section 3, we introduce the related work. The model for Bayesian cluster ensemble is given in Section 4 and two inference algorithms—variational inference and Gibbs sampling for BCE—are provided in section 5 and 6 respectively. We report experimental results in section 7, and conclude in section 8.

## 2 Problem Formulation

Given  $N$  objects  $O = \{\mathbf{o}_i, [i]_1^N\}$  ( $[i]_1^N \equiv i = 1 \dots N$ ) and  $M$  base clustering algorithms  $C = \{c_j, [j]_1^M\}$ , we get  $M$  base clusterings of the objects, 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  $N$  objects  $\{\mathbf{o}_i, [i]_1^N\}$ . 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  $c_j$ , then 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  $j^{\text{th}}$  column is  $\lambda_j$ , as shown in Figure 1(a). The matrix can be viewed from another perspective: Each row  $\mathbf{x}_i$  of the matrix, i.e., all base clustering results for  $\mathbf{o}_i$ , gives a new feature vector representation of the object  $\mathbf{o}_i$  (Figure 1(b)). 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  $N$  objects 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” (Figure 1(a)), i.e., 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. The cluster correspondence problem is hard to solve efficiently, and the complexity increases especially when different base clustering algorithms generate different number of clusters [21].

A simpler approach to cluster ensemble problem, which is what we use in this paper, is to read the matrix  $B$  in a “row-wise” (Figure 1(b)) way. All base clustering results for an object  $\mathbf{o}_i$  can be considered as a feature vector  $\mathbf{x}_i$  with discrete feature values [21], and we consider base clustering matrix  $B$  as a set of  $N$  rows of  $M$ -dimensional feature vectors  $\{\mathbf{x}_i, [i]_1^N\}$ . From this perspective, the cluster ensemble problem becomes the one of finding a clustering  $\lambda^*$  for feature vectors  $\{\mathbf{x}_i, [i]_1^N\}$ , 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.

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

(a)

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

(b)

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

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

**2.1 Missing value cluster ensembles.** When several base clustering results are missing for several objects, we have a missing value cluster ensemble problem. Such a problem appears due to various reasons. For example, if there are new objects added to the dataset after running clustering algorithm  $c_j$ , these new objects 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 objects (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 [10], 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. Meanwhile, combining the results on different subsets helps to generate a more reasonable ensemble clustering. Therefore, it is desirable to learn a consensus clustering in a row-distributed manner.

**2.3 Column-distributed cluster ensemble.** For column-distributed cluster ensemble, different base clustering results of all objects are at different locations. The corresponding real life scenario is that separate organizations have different base clusterings on the same set of objects, e.g., 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.

### 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 [20, 7, 1, 6]. 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.

In particular, Strehl *et al.* [20] presents three graph-based cluster ensemble algorithms. The cluster-based similarity partitioning algorithm (CSPA) [20] induces a graph from a co-association matrix, and the graph is partitioned by the METIS algorithm [12] to obtain final clusters. In addition, Hypergraph partitioning algorithm (HGPA) [20] represents each cluster and corresponding objects by a hyperedge and nodes respectively, and then minimal cut algorithm HMETIS [11] is applied for partitioning. Further, hyperedge collapsing operations are used in a meta-clustering algorithm (MCLA) [20] which determines a soft cluster membership for each object.

Fern *et al.* [7] propose a bipartite graph partitioning algorithm, which 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 *et al.* [1] 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 [8, 15, 13, 16]. The main idea of this category is converting base clustering matrix into another matrix such as co-association matrix, consensus matrix or nonnegative matrix, and using matrix operations to get the results of cluster ensemble.

Fred *et al.* [8] 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.* [13] 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.* [16] 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.

In the paper [15], Jordan *et al.* illustrate that the problem of cluster ensemble can be formulated under the framework of nonnegative matrix factorization (NMF), which refers to the problem of factorizing a given nonnegative data matrix  $X$  into two matrix factors, i.e.,  $X \approx AB$ , under the constraint of  $A$  and  $B$  to be nonnegative matrices.

**3.3 Probabilistic models.** The third class of cluster ensemble algorithms are based on probabilistic models [21]. The algorithms take advantage of statistic properties of base clusterings results to achieve a consensus clustering. Topchy *et al.* [21] consider a representation of multiple clusterings as a set of new attributes characterizing the data items, and a mixture model offers a probabilistic model of consensus using a finite mix-

ture 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 algorithm.

#### 4 Bayesian Cluster Ensembles

In this section, we propose a novel Bayesian Cluster Ensemble (BCE) model. The main idea is as follows: Given a base clustering matrix  $B$  as a group of feature vectors  $\{\mathbf{x}_i, [i]_1^N\}$ , we assume there exists a Bayesian graphical model generating  $B$ . In particular, we assume that each object  $\mathbf{x}_i$  has an underlying mixed-membership to different consensus clusters. Let  $\boldsymbol{\theta}_i$  denote the latent mixed-membership vector for  $\mathbf{x}_i$ ; if there are  $k$  consensus clusters,  $\boldsymbol{\theta}_i$  is a discrete distribution over the  $k$  clusters. From the generative model perspective, we assume that  $\boldsymbol{\theta}_i$  is sampled from a Dirichlet distribution, with parameter  $\alpha$ . Further, each latent consensus cluster  $h, [h]_1^k$ , has a discrete distribution  $\beta_{hj}$  over the cluster *ids*  $\{1, \dots, k_j\}$  for base clustering  $c_j$ . Thus, if an object truly belongs to consensus cluster  $h$  for  $c_j$ , then its cluster *id*  $x_{ij} = r \in \{1, \dots, k_j\}$  according to base clustering  $c_j$  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:

1. Choose  $\boldsymbol{\theta}_i \sim \text{Dirichlet}(\alpha)$ .
2. For the  $j^{\text{th}}$  base clustering:
  - (a) Choose a component  $z_{ij} = h \sim \text{discrete}(\boldsymbol{\theta}_i)$ ;
  - (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  $(\boldsymbol{\theta}_i, z_{ij})$  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 [2, 19] by choosing a discrete distribution as the generative model. Further, BCE is closely related to LDA [4], 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  $\{\mathbf{x}_i, \mathbf{z}_i, \boldsymbol{\theta}_i\}$  is 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  $j^{\text{th}}$  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:

$$(4.1) \quad p(\mathbf{x}_i | \alpha, \beta) = \int_{\boldsymbol{\theta}_i} p(\boldsymbol{\theta}_i | \alpha) \prod_{j=1, \exists x_{ij}}^M \sum_h p(z_{ij} = h | \boldsymbol{\theta}_i) p(x_{ij} | \beta_{hj}) d\boldsymbol{\theta}_i.$$

#### 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. Expectation maximization (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

$$(5.2) \quad p(\boldsymbol{\theta}_i, \mathbf{z}_i | \mathbf{x}_i, \alpha, \beta) = \frac{p(\boldsymbol{\theta}_i, \mathbf{z}_i, \mathbf{x}_i | \alpha, \beta)}{p(\mathbf{x}_i | \alpha, \beta)},$$

cannot be calculated in closed form since the denominator (partition function)  $p(\mathbf{x}_i)$  as an expansion of (4.1) 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  $j^{\text{th}}$  base clustering assigns  $\mathbf{o}_i$  to base cluster  $r$  and 0 otherwise,  $\beta_{hj}(r)$  is the  $r^{\text{th}}$  component of the discrete distribution  $\beta_{hj}$  for the  $h^{\text{th}}$  consensus cluster and the  $j^{\text{th}}$  base clustering. The coupling between  $\theta$  and  $\beta$  in the summation over the latent variable  $z$  makes the computation intractable [4]. There are two main classes of approximation algorithms to address such problems: one is variational inference, and the other is Gibbs sampling. In this section, we present the variational inference method, and in the next section, a Gibbs sampling method is presented.

**5.1 Variational inference.** Since it is intractable to calculate the true posterior (5.2) 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 [2, 4], we introduce a family of

variational distributions as

$$(5.3) \quad q(\boldsymbol{\theta}_i, \mathbf{z}_i | \gamma_i, \phi_i) = q(\boldsymbol{\theta}_i | \gamma_i) \prod_{j=1}^M q(z_{ij} | \phi_{ij})$$

as an approximation of  $p(\boldsymbol{\theta}_i, \mathbf{z}_i | \alpha, \beta, \mathbf{x}_i)$  in (5.2), 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  $L(\alpha, \beta; \phi_i, \gamma_i)$  on  $\log p(\mathbf{x}_i | \alpha, \beta)$  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

$$(5.4) \quad \log p(B | \alpha, \beta) = \sum_{i=1}^N \log p(\mathbf{x}_i | \alpha, \beta) \geq \sum_{i=1}^N L(\alpha, \beta; \phi_i, \gamma_i).$$

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 possible bound from this family of approximations. A direct calculation leads to the following set of update equations for the variational maximization:

$$(5.5) \quad \phi_{ijh} \propto \exp\left(\Psi(\gamma_{ih}) - \Psi\left(\sum_{h'} \gamma_{ih'}\right)\right) + \sum_{r=1}^{k_j} \mathbb{1}(r|i, j) \log \beta_{hj}(r)$$

$$(5.6) \quad \gamma_{ih} = \alpha_h + \sum_{j=1}^M \phi_{ijh},$$

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 (5.4) is maximized by the point estimate for  $\beta$ :

$$(5.7) \quad \beta_{hj}(r) \propto \sum_{i=1}^N \phi_{ijh} \mathbb{1}(r|i, j),$$

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 [4]. In particular, the update equation for  $\alpha_h$  is given by

$$(5.8) \quad \alpha'_h = \alpha_h - \frac{g_h - c}{l_h},$$

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, i.e., the first derivative of the log Gamma function.

**5.2 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:

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

$L(\alpha, \beta, \phi_i^{(t)}, \gamma_i^{(t)})$  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:

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

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,

$$(5.9) \quad \sum_{i=1}^N L(\alpha^{(t-1)}, \beta^{(t-1)}, \phi_i^{(t-1)}, \gamma_i^{(t-1)}) \leq \sum_{i=1}^N L(\alpha^{(t-1)}, \beta^{(t-1)}, \phi_i^{(t)}, \gamma_i^{(t)})$$

$$(5.10) \quad \leq \sum_{i=1}^N L(\alpha^{(t)}, \beta^{(t)}, \phi_i^{(t)}, \gamma_i^{(t)}).$$

The first inequality holds because in the E-step, (5.9) 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, (5.10) is the maximum of  $L(\alpha, \beta, \phi_i^{(t)}, \gamma_i^{(t)})$ . Therefore, the objective function is non-decreasing until convergence [17].

**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 (5.5) and (5.6), 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 (5.7), we can write the right term of (5.7) 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 (5.7) 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 equation (5.8) 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 (5.8). 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 (5.8). 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 (5.5) and (5.6). Also, we update  $(\alpha_{(1)}, \beta_{(1)})$  to get  $(\alpha_{(1)}^{(t)}, \beta_{(1)}^{(t)})$  following (5.8) and (5.7) 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 (5.5) and (5.6). 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 are 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  $(\alpha_j^{(t)}, \beta_j^{(t)})$  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  $\beta_j^{(t)}$  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  $[i]_1^N$ ,  $[h]_1^N$  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 (5.5). 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 (5.6). 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 (5.7) 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 (5.8) 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  $(\alpha^{(t-1)}, \beta^{(t-1)})$ , i.e., 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.

## 6 Gibbs Sampling for BCE

In this section, we propose a Gibbs sampling [5] method to obtain the posterior distribution by sampling. Gibbs sampling is a special case of Markov chain Monte Carlo (MCMC) [18], and forms a powerful stochastic approximation method for inference in graphical models. To leverage existing results in the Gibbs sampling literature [9], we mildly modify the BCE model outlined in Section 4, by introducing a prior distribution Dirichlet( $\omega_j$ ) for each  $\beta_j, [j]_1^M$ . The generative model for BCE could then be represented as follows:

$$\begin{aligned} \theta_i &\sim \text{Dirichlet}(\alpha), \\ \mathbf{z}_i | \theta_i &\sim \text{discrete}(\theta_i), \\ \beta_j &\sim \text{Dirichlet}(\omega_j), \\ x_{ij} | z_{ij}, \beta_j &\sim \text{Dirichlet}(\beta_j), \end{aligned}$$

where  $[i]_1^N, [j]_1^M$ . For simplicity, here we assume Dirichlet( $\alpha$ ) and Dirichlet( $\omega_j$ ) are symmetric Dirichlet priors, which are fixed upfront.

Dataset	Instances	Features	Categories
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

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

The conditional posterior distribution for  $z_{ij}$  is given by

$$(6.11) \quad \begin{aligned} P(z_{ij} = h | z_{-(ij)}, x_{ij}) \\ \propto P(x_{ij} | z_{ij} = h, z_{-(ij)}, x_{-(ij)}) p(z_{ij} = h | z_{-(ij)}). \end{aligned}$$

The first term of (6.11) is given by

$$P(x_{ij} | z_{ij} = h, z_{-(ij)}, x_{-(ij)}) = \frac{\omega_j + n_{-(ij),h}^{(r)}}{k_j \omega_j + n_{-(ij),h}^{(\cdot)}},$$

where  $n_{-(ij),h}^r$  is the number of instances of the  $j^{th}$  base clustering results  $\{x_{(\cdot j)} = r, [r]_1^{k_j}\}$  assigned to the consensus cluster  $h$ , not including the current one,  $n_{-(ij),h}^{(\cdot)}$  is the total number of  $x_{ij}$  assigned to the consensus cluster  $h$ , not including the current one. The second term of (6.11) is given by

$$p(z_{ij} = h | z_{-(ij)}) = \frac{\alpha + n_{-(ij),h}}{k\alpha + n_{-(ij)}},$$

where  $n_{-(ij),h}$  is the number of  $x_{ij}$  from object  $i$  assigned to consensus cluster  $h$ , not including the current one, and  $n_{-(ij)}$  is the total number of  $x_{ij}$  from object  $i$ , not including the current one, and from above two equations, we can get the following formula

$$P(z_{ij} = h | z_{-(ij)}, x_{ij}) \propto \frac{\omega_j + n_{-(ij),h}^r}{k_j \omega_j + n_{-(ij),h}^{(\cdot)}} \frac{\alpha + n_{-(ij),h}}{k\alpha + n_{-(ij)}}.$$

## 7 Experimental Results

In this section, we run experiments on ten datasets from UCI machine learning repository. The numbers of objects, features and classes in each data set are listed in Table 1. For all reported results, there are two steps leading to the final consensus clustering. First, we use  $k$ -means with different initializations to obtain a set of (20, unless otherwise mentioned) base clusterings. Second, various cluster ensemble algorithms, including mixture model (MM) [21], CSPA, HGPA, MCLA [20] and

Algorithm	general	miss-v	increase-c	column-d	row-d
K-MEANS	✓	×	✓	×	✓
CSPA	✓	✓	✓	×	×
HGPA	✓	✓	✓	×	×
MCLA	✓	✓	✓	×	×
MM	✓	✓	✓	✓	✓
BCE	✓	✓	✓	✓	✓

Table 2: The applicability of algorithms to different experimental settings: ✓ indicates that the algorithm is applicable, and × indicates otherwise.

$k$ -Means, are applied to the base clustering results to generate a consensus clustering. We compare their results with BCE, which uses both variational approximation and Gibbs sampling for inference, represented as V-BCE and G-BCE respectively.

Experiments 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).
4. Column-distributed cluster ensemble (column-d).
5. Row-distributed cluster ensemble (row-d).

Table 2 shows the five categories of experiments above and the six 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 seen from Table 2, BCE is the most flexible and versatile among the six algorithms.

For evaluation, we use micro-precision [22] to measure accuracy of the consensus cluster with respect to the true labels: the micro-precision  $MP = \sum_{h=1}^k a_h/n$ , 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.

**7.1 General cluster ensembles.** We present results for the five categories of experiments listed in Table 2, starting from general cluster ensemble in this subsection.

Given  $N$  objects, by running  $k$ -means 2000 times, we obtain 2000 base clustering results, which are divided evenly into 100 subsets, with an  $N \times 20$  base clustering matrix each. We run each cluster ensemble algorithm 100 times on each of these 100 base clustering matrices. The maximum and average  $MP$ s are reported in Table 3. The key observations can be summarized as follows: (1) V-BCE and G-BCE almost always have 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. (2) BCE outperforms other cluster ensemble algorithms for most of the times. In terms of the maximum  $MP$ , V-BCE wins eight out of ten times, whereas CSPA and MM win once each. In terms of the average  $MP$ , V-BCE wins five out of ten times, CSPA wins three times, G-BCE wins twice, and MM wins once. Since the results of MM and V-BCE are rather close to each other, to make a careful comparison, we run a paired  $t$ -test under the hypothesis

$$\begin{aligned}
 H_0 &: MP(\text{MM}) = MP(\text{V-BCE}) \\
 H_a &: MP(\text{MM}) < MP(\text{V-BCE}) .
 \end{aligned}$$

The results are shown in Table 4. V-BCE outperforms MM nine out of ten times with a low  $p$ -value ( $< 0.05$ ) most of the times, indicating that  $MP(\text{V-BCE})$  is significantly better than  $MP(\text{MM})$  on these datasets. In addition, the smaller standard deviation of V-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. We compare the performance of different algorithms except  $k$ -means, because  $k$ -means cannot handle missing values. Each time we run the algorithms 10 times and report  $MP$  in Figure 2. 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.

### 7.3 Cluster ensembles with increasing columns.

In order to find out how increasing the number of base clusterings affects the cluster ensemble accuracy, 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

dataset \ algorithms	The results of base clusterings K-means		MCLA		CSPA		HGPA		MM		K-means cluster ensemble		G-BCE		V-BCE random initialization	
	Max	average	Max	average	Max	average	Max	average	Max	average	Max	average	Max	average	Max	average
iris	0.8867	0.6267	0.8867	0.8867	0.9533	<b>0.9167</b>	0.7333	0.7333	0.9067	0.8867	0.5267	0.5267	0.9533	0.8697	<b>0.9600</b>	0.8911
wdbc	0.8541	0.7595	0.8840	0.8840	0.8840	0.8840	0.5518	0.5188	0.8840	0.8840	0.8840	0.8689	<b>0.8893</b>	<b>0.8893</b>	<b>0.8893</b>	0.8840
ionosphere	0.7123	0.6906	0.7123	0.7046	0.6952	0.6952	0.6353	0.6063	0.7179	0.7111	0.7094	0.7094	0.7236	0.7073	<b>0.7749</b>	<b>0.7123</b>
glass	0.5421	0.5140	0.5187	0.4766	0.4393	0.4393	0.4439	0.4234	0.5748	0.5519	0.5093	0.4363	0.5514	0.4867	<b>0.6121</b>	<b>0.5526</b>
bupa	0.4841	0.4537	0.5652	0.5652	0.5710	<b>0.5710</b>	0.5188	0.5075	0.5710	0.5586	0.5565	0.5164	0.5710	<b>0.5710</b>	<b>0.5942</b>	0.5664
pima	0.6602	0.5751	0.6602	0.6602	0.5065	0.5065	0.5260	0.5163	0.6654	0.6503	0.6029	0.6029	0.6615	0.6445	<b>0.7044</b>	<b>0.6612</b>
wine	0.6629	0.5904	0.7247	0.7247	<b>0.7416</b>	<b>0.7416</b>	0.5562	0.5250	0.7247	0.7129	0.4775	0.4775	0.6966	0.6559	0.7247	0.7247
magic04	0.6491	0.6252	0.6491	0.6491	×	×	0.6491	0.6235	0.6530	0.6231	0.6491	0.6250	0.6491	0.6491	<b>0.6531</b>	<b>0.6497</b>
balance	0.5936	0.5114	0.5216	0.5188	0.5408	0.5408	0.4256	0.4256	<b>0.6016</b>	<b>0.5514</b>	0.5824	0.5824	0.5714	0.5150	0.5968	0.5293
segmentation	0.5710	0.5574	0.5657	0.5657	0.5810	0.5810	0.5419	0.4543	0.6233	0.5817	0.5710	0.5142	0.5233	0.5233	<b>0.6362</b>	<b>0.5854</b>

Table 3: Maximum and average  $MP$  on different datasets by running different cluster ensemble algorithms. (Magic04 is too large such that CSPA could not finish its run).

Dataset	Mean-D	sd-MM	sd-V-BCE	p-value
pima	<b>-0.0117</b>	0.0205	<b>0.0038</b>	0.0089
iris	<b>-0.0402</b>	0.0221	<b>0.0103</b>	0.0026
wdbc	<b>-0.0009</b>	0.0018	<b>0.0000</b>	<b>0.3256</b>
balance	0.0301	0.0384	<b>0.0061</b>	0.0010
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
wine	<b>-0.0240</b>	0.0290	<b>0.0119</b>	0.0239
magic04	<b>-0.0010</b>	0.0023	<b>0.0014</b>	<b>0.4127</b>
ionosphere	<b>-0.0013</b>	0.0024	<b>0.0000</b>	0.0169
segmentation	<b>-0.0140</b>	0.0331	<b>0.0186</b>	0.2250

Table 4: The paired  $t$ -test of  $MP$  from MM and BCE. “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).

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 Figure 3.

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 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 [10]. 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 10 iterations. The comparative results are presented in Figure 4. 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 the fact 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 Figure 5. 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 vari-

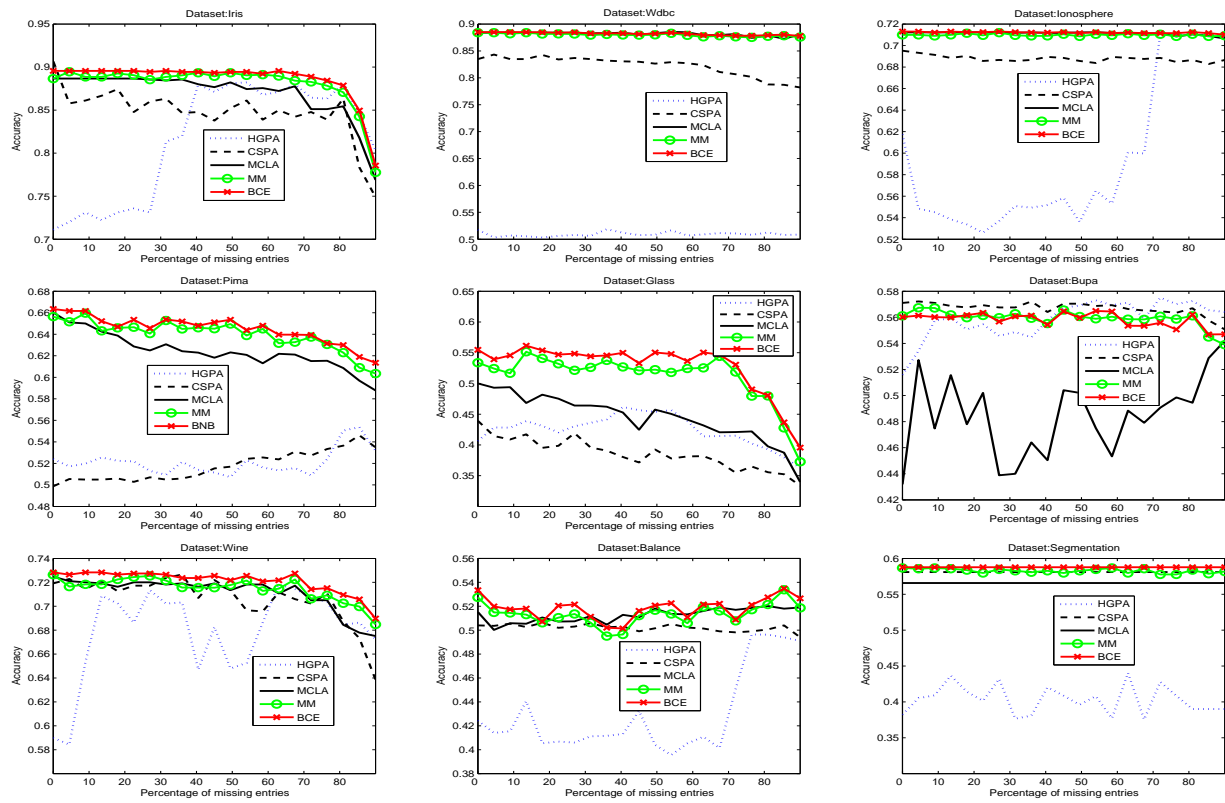


Figure 2: Average  $MP$  with increasing percentage of missing values.

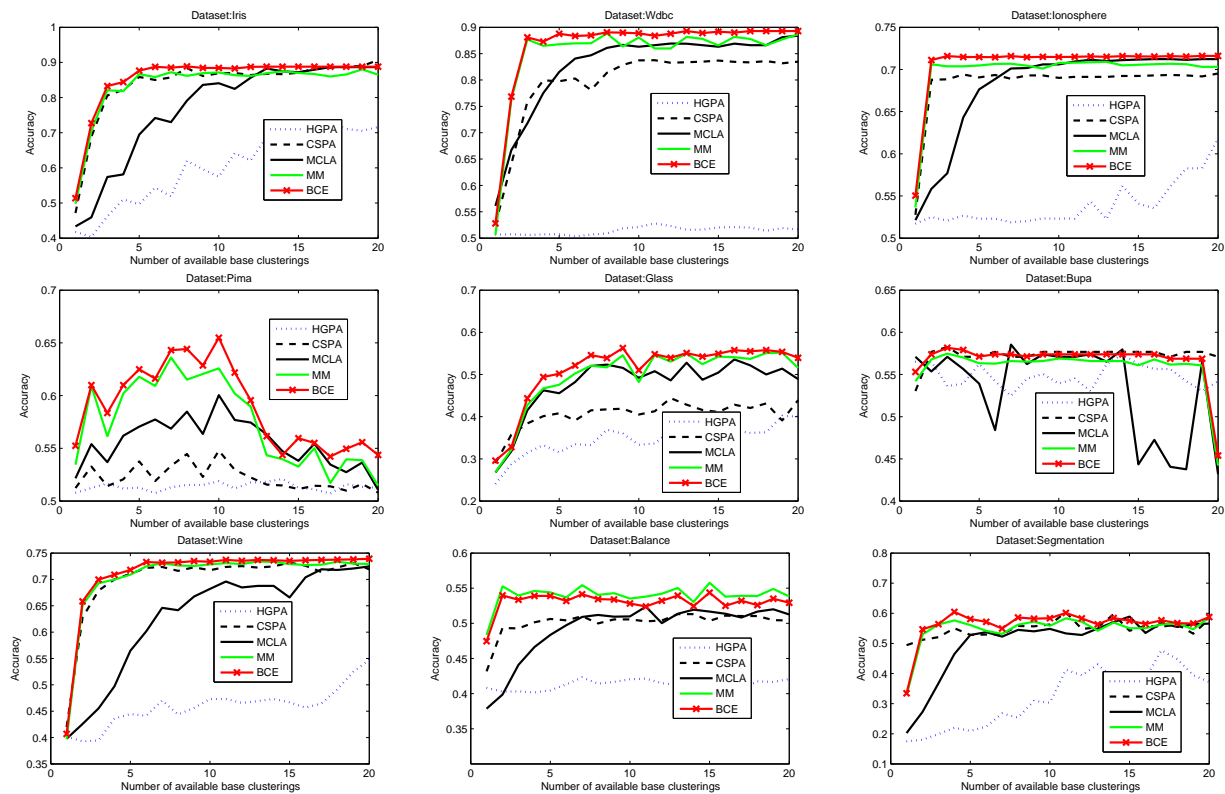


Figure 3: Average  $MP$  comparison with increasing number of available base clusterings.

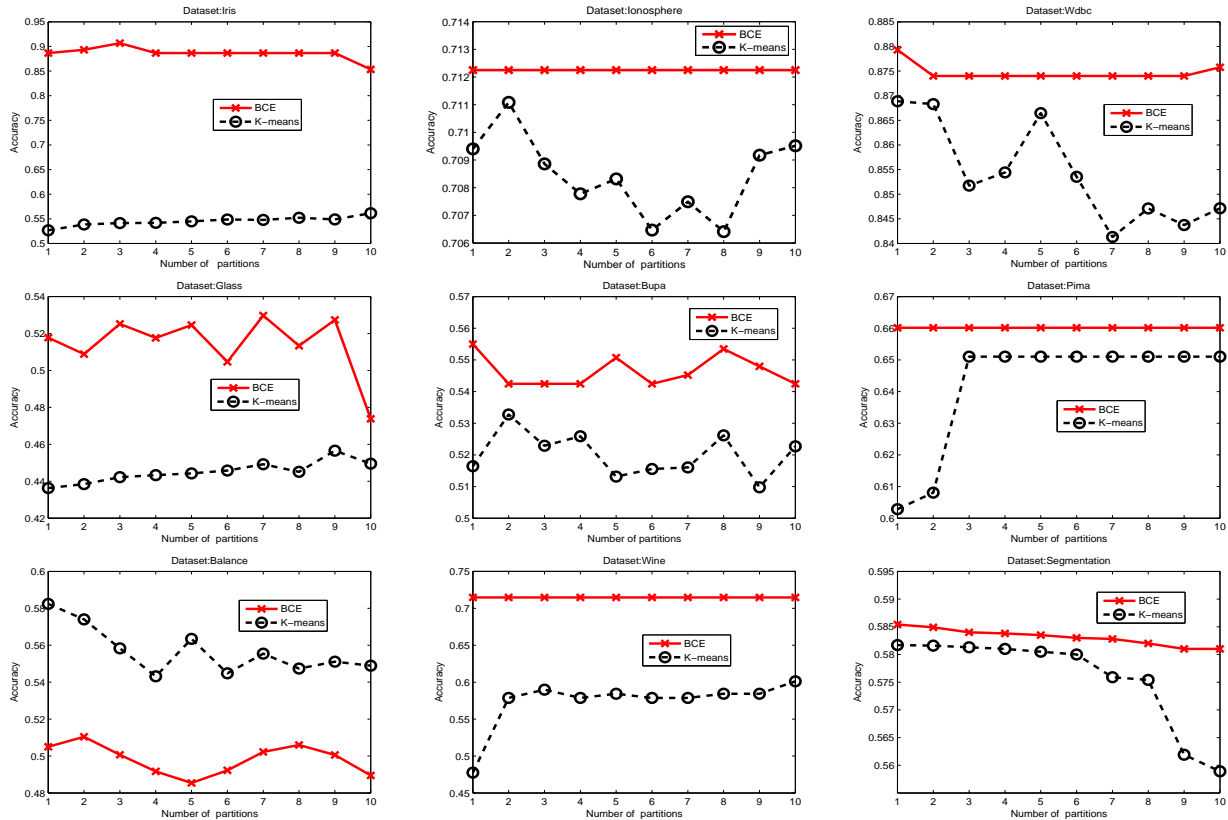


Figure 4: Average  $MP$  with increasing number of distributed partitions.

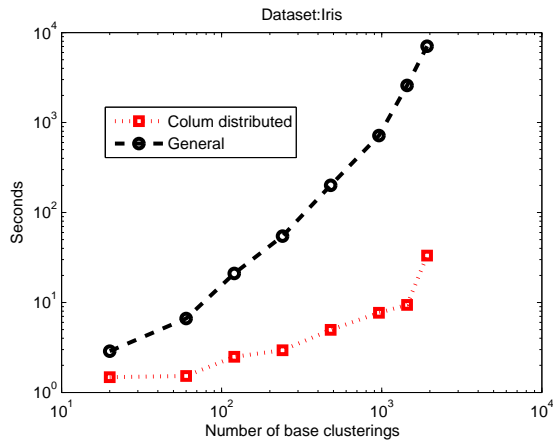


Figure 5: The comparison of running time between column-distributed and general cluster ensemble.

ant is several orders of magnitudes faster. Therefore, the column-distributed BCE is readily applicable to the real life settings with large data sets.

## 8 Conclusion

In this paper, we have proposed Bayesian cluster ensembles (BCE), a mixed-membership generative model for obtaining a consensus clustering by combining mul-

iple 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. We have proposed two methods, respectively based on variational approximation and Gibbs sampling, for learning a Bayesian cluster ensemble. 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. Finally, the proposed algorithms for BCE can be run in a distributed manner without exchanging base clustering results, thereby preserving privacy and/or substantial speed-ups.

**Acknowledgements:** The research was supported by NASA grant NNX08AC36A and NSF grant IIS-0812183.

## References

- [1] M. Al-Razgan and C. Domeniconi. Weighted cluster ensemble, In *SDM*, pp. 258–269, 2006.

- [2] A. Banerjee and H. Shan. Latent Dirichlet conditional naive-Bayes models, In *ICDM*, pp. 421–426, 2007.
- [3] J. Barthelemy and B. Leclerc. The median procedure for partition, In *Partitioning Data Sets, I.J. Cox et al eds., AMS DIMACS Series in Discrete Mathematics*, pp. 3–34, 1995.
- [4] D. Blei, A. Ng, and M. Jordan. Latent Dirichlet allocation, *JMLR*, 3:3993–1022, 2003.
- [5] G. Casella and E. George. Explaining the Gibbs sampler, *The American Statistician*, 46:167–174, 1992.
- [6] X. Fern and C. Brodley. Random Projection for High Dimensional Data Clustering: A Cluster Ensemble Approach, In *ICML*, pp. 186–193, 2003
- [7] X. Fern and C. Brodley. Solving cluster ensemble problems by bipartite graph partitioning, In *ICML*, pp. 281–288, 2004.
- [8] A. Fred and A. Jain. Data clustering using evidence accumulation, In *ICPR*, pp. 276–280, 2002.
- [9] T. Griffiths and M. Steyvers Finding scientific topics, *PNAS*, pp.5228-5235, 2004.
- [10] G. Jagannathan and R. Wright. Privacy-preserving distributed k-means clustering over arbitrarily partitioned data, In *KDD*, pp. 593–599, 2005.
- [11] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar. Multilevel hypergraph partitioning: Applications in VLSI design, In *ACM/IEEE Design Automation Conference*, pp. 526–529, 1997.
- [12] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs, *SIAM Journal on Scientific Computing*, 20(1):359 – 392, 1999.
- [13] P. Kellam, X. Liu, N. Martin, C. Orengo, S. Swift, and A. Tucker. Comparing, contrasting and combining clusters in viral gene expression data, *Workshop on Intelligent Data Analysis in Medicine and Pharmacology*, pp. 56–62, 2001.
- [14] L. Kuncheva and D. Vetrov. Evaluation of stability of k-means cluster ensembles with respect to random initialization, *PAMI*, 28(11):1798–1808, 2006.
- [15] T. Li, C. Ding, and M. Jordan. Solving consensus and semi-supervised clustering problems using nonnegative matrix factorization, In *ICDM*, 2007.
- [16] 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, *Machine Learning Journal*, 52:91–118, 2003.
- [17] R. Neal and G. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. *Learning in Graphical Models*, pp.355-368, 1998
- [18] C. Robert and G. Casella. *Monte Carlo statistical methods (second edition)*, 2004.
- [19] H. Shan and A. Banerjee. Mixed-membership naive Bayes models, *Technical Report*, TR 09-002, Department of Computer Science and Engineering, University of Minnesota, Twin Cities, 2009.
- [20] A. Strehl and J. Ghosh. Cluster ensembles - a knowledge reuse framework for combining multiple partitions, *JMLR*, 3:583–617, 2002.
- [21] A. Topchy, A. Jain, and W. Punch. A mixture model for clustering ensembles, In *SDM*, pp. 379–390, 2004.
- [22] Z. Zhou and W. Tang. Clusterer ensemble, *Knowledge-Based Systems*, 1(19):77–83, 2006.