Advances in Fuzzy Clustering and Its Applications

J. Valente de Oliveira and W. Pedrycz (Editors)

Contents

Preface

Cluster Ensembles is a framework for combining multiple partitionings obtained from separate clustering runs into a final consensus clustering. This framework has attracted much interest recently because of its numerous practical applications, and a variety of approaches including Graph Partitioning, Maximum Likelihood, Genetic algorithms, and Voting-Merging have been proposed. The vast majority of these approaches accept hard clusterings as input. There are, however, many clustering algorithms such as EM and fuzzy c-means that naturally output soft partitionings of data, and forcibly hardening these partitions before obtaining a consensus potentially involves loss of valuable information. In this chapter we propose several consensus algorithms that accept soft clusterings and experiment over many real-life datasets to show, both conceptually and empirically, that using soft clusterings as input does offer significant advantages, especially when dealing with vertically partitioned data.

The authors wish to thank Ray Mooney, Srujana Merugu, Arindam Banerjee, Suju Rajan, and Sreangsu Acharyya for helpful discussions and thoughtful suggestions through the course of this work.

1

Soft Cluster Ensembles

Kunal Punera and Joydeep Ghosh

Dept. of Electrical and Computer Engineering,
University of Texas at Austin

1.1 Introduction

Cluster Ensembles is a “Knowledge Reuse” framework for combining multiple clusterings of a set of objects without accessing the original features of the objects. This problem was first proposed in (Strehl and Ghosh 2002) where the authors applied it for improving the quality and robustness of clustering, and in distributed clustering. A related problem of consensus clustering also exists in the marketing literature (Kreiger and Green 1999) where often a set of individuals are segmented in multiple ways based on different criteria (needs-based, demographics, etc), and one is interested in obtaining a single, unified segmentation.

The idea of combining multiple models is well established in the classification and regression scenarios where it has led to impressive improvements in a wide variety of domains (Breiman 1999; Freund and Schapire 1996; Ghosh 2002). Combining clusterings is, however, a more difficult problem than combining the results of multiple classifiers, since clusterings are invariant to cluster label permutations. In other words, all partitions of a set of objects that differ only in the cluster labeling are identical. As a result, before combining the clusterings one has to identify which clusters from different clusterings correspond to each other. This sub-problem of identifying cluster correspondences is further complicated by the fact that the number of clusters in the individual solutions might vary significantly. These differences along with wide variations in the clustering algorithms and features of data used for underlying clustering algorithms make solving cluster ensembles a very challenging problem. Even

Advances in Fuzzy Clustering and Its Applications. Edited by J. Valente de Oliveira and W. Pedrycz
© 2001 John Wiley & Sons, Ltd

so, the ability to combine clusterings in an ensemble is very useful.

Cluster ensembles have been shown to be useful in many application scenarios. Some of the principal ones are:

- **Knowledge reuse**: An important application of cluster ensembles is combining knowledge encoded in multiple clusterings. An example of this is exploiting the knowledge in legacy clusterings while re-clustering the data. We might not have access to the features that were originally used while creating the legacy clusterings; they might even have been created manually by a domain expert. Also, in many cases the number of clusters in the original data might have changed or new features might now be available. In these cases re-clustering all the data with the new features may not be possible. Cluster ensembles can be employed to combine multiple clusterings in these feature/object distributed scenarios (Ghosh et al. 2002; Strehl and Ghosh 2002).

- **Multi-View clustering**: A set of objects can be clustered multiple times using different attributes/criteria. For example, in marketing applications, customers can be segmented based on their needs, psycho-graphic or demographic profiles, brand choices, etc. Consensus clustering can be used to combine all such partitions into one, which is often easier to act on (Kreiger and Green 1999).

- **Distributed computing**: In many applications, the data to be clustered is distributed over many sites, and data sharing is prohibited. In the case of distributed computing, communication costs make sharing all the data with a central site prohibitively expensive, but communicating clustering results is cheaper. In other cases, while sharing actual features of data might be prohibited because of privacy reasons, the sharing of clustering results might be permissible, as in (Merugu and Ghosh 2003). Both these scenarios can be handled by locally clustering data present at each site, and then transferring only the clustering solutions to a central site. Cluster ensemble algorithms can then be used to combine these clusterings into a composite clustering at the central site.

- **Improved quality of results**: Each clustering algorithm has its own search biases and uses different types of features. Combining the results of multiple different clusterings algorithms could give improvements over their individual solutions, as the combined solution would take into account all their biases. It has been seen that using cluster ensembles to combine diverse clustering solutions leads to more accurate results on average (Hadjitodorov et al. 2006; Kuncheva and Hadjitodorov 2004).

- **Robust solutions**: Many clustering algorithms suffer from initialization problems, often finding local minima of their objective functions. The cluster ensembles framework can be used to alleviate these problems of unstable clustering results. Multiple runs of a clustering algorithm, obtained with different initializations or with different sets of features, can be combined in order to obtain a robust final solution (Fern and Brodley 2003; Fred and Jain 2002).

There have been several attempts to solve cluster ensembles in the recent past. Strehl and Ghosh (2002) proposed three graph-theoretic approaches for finding the consensus clustering. A bipartite graph partitioning based approach has been proposed by Fern and Brodley (2004). Topchy et al. (2004) proposed the use of mixture of multinomial distributions to model the ensemble of labels along the lines of classical latent class analysis in marketing literature. Some of these approaches will be described in detail in Section 1.2. While these techniques are very varied in the algorithms they employ, there is a common thread that they only work with hard constituent clusterings. It is the goal of this chapter to investigate *Soft* Cluster Ensembles.

1.1.1 Ensembles of Soft Clusterings

There are several clustering algorithms, such as EM (Dempster et al. 1977) and fuzzy c-means (Bezdek and Pal 1992; Dunn 1973), that naturally output soft partitions of data. A soft partition assigns a value for the degree of association of each instance to each output cluster. So instead of a label vector for all the instances we have a matrix of values in which each instance is associated with every cluster with some membership value; often these values are the posterior probabilities and sum upto to one. In order to solve an ensemble formed of soft clusterings using one of the existing algorithms mentioned above, we would have to "harden" the clusterings. This process involves completely assigning each instance to the cluster to which it is most associated. This process results in the loss of the information contained in the uncertainties of the cluster assignments. This is especially true for application settings where underlying clustering algorithms access partial views of the data, such as in distributed data mining. A landmark work on "collaborative" fuzzy clustering was done by Pedrycz (2002). The author considered a vertical partitioning scenario, and captured the collaboration between multiple partitionings via pair wise interaction coefficients. This resulted in an extended cost function to accommodate the collaboration effect in the optimization process. This approach is restricted in scope in many ways: each partition needs to have the same number of clusters; the difficult cluster correspondence problem is assumed to be already solved; and the distances between each point and its representative in each of the solutions need to be known. Despite these constraints, it was illustrated that, at least for simple 2 and 3 cluster problems, collaboration had a positive effect on cluster quality. This further motivates the present study, where we propose flexible frameworks for combining multiple soft clusterings directly without "hardening" the individual solutions first. We introduce a new consensus function (ITK) based on the Information-Theoretic KMeans algorithm (Dhillon et al. 2003b) that is more efficient and effective than existing approaches. For evaluation purposes, we create a large number of ensembles of varying degrees of difficulty, and report clustering results achieved by the various existing and new algorithms on them. In order to objectively evaluate ITK we extend existing algorithms to operate on soft cluster ensembles as well.

1.1.2 Organization of this Chapter

In Section 1.2 we first define the *hard* cluster ensemble problem formally, and then go on to describe the various consensus functions that have been proposed in literature. The *soft* cluster ensembles are then formally introduced in Section 1.3 followed by several new consensus functions which operate on them. The experimental setup for our extensive evaluation of these algorithms and the empirical results then follow in Section 1.4 and Section 1.5 respectively. Finally, in Section 1.6 we conclude the chapter and briefly mention possible directions for future research.

1.2 Cluster Ensembles

In this section, we will first define the *hard* clusters ensembles problem formally, and then present graph-theoretic solutions proposed by Strehl and Ghosh (2002) and Fern and Brodley (2004). We will also present some related work on robust clustering by Fred and Jain (2002) and on generative models for ensembles by Topchy et al. (2004). Other methods such as Voting-Merging (Dimitriadou et al. 2001) and GA-Search (Gablentz et al. 2000) are not presented as they are either not competitive or too restrictive in their scope. We will end the section with a brief discussion on past work on role of diversity in the cluster ensembles problem.

Table 1.1 A set of three clusterings

	$\lambda^{(1)}$	$\lambda^{(2)}$	$\lambda^{(3)}$
x_1	1	2	1
x_2	1	2	1
x_3	1	3	2
x_4	2	3	2
x_5	2	3	3
x_6	3	1	3
x_7	3	1	3

Table 1.2 Hyper-graph representation of clusterings

	$H^{(1)}$			$H^{(2)}$			$H^{(3)}$		
	h_1	h_2	h_3	h_4	h_5	h_6	h_7	h_8	h_9
v_1	1	0	0	0	1	0	1	0	0
v_2	1	0	0	0	1	0	1	0	0
v_3	1	0	0	0	0	1	0	1	0
v_4	0	1	0	0	0	1	0	1	0
v_5	0	1	0	0	0	1	0	0	1
v_6	0	0	1	1	0	0	0	0	1
v_7	0	0	1	1	0	0	0	0	1

1.2.1 The Hard Cluster Ensemble problem

Let $X = \{x_1, x_2, ..., x_n\}$ denote a set of instances/objects. Each partitioning of the data (called a clustering) is represented as a vector of labels over the data. Let $\lambda^{(q)} \in \{1, 2, ...k^{(q)}\}^n$ denote the label vector of the q^{th} constituent clustering of X; *i.e.* $\lambda_i^{(q)}$ is the label of x_i in the q^{th} partitioning. A set of r such clusterings $\lambda^{(1,2,\ldots,r)}$ is called a cluster ensemble (for an example, see Table 1.1). The goal is to find a consensus function Γ which would combine the r clusterings $\lambda^{(1,2,\ldots,r)}$ into a single clustering/labeling λ.

It is instructive, for presentation later in this section, to consider that every hard clustering can be mapped to a hyper-graph. A hyper-graph consists of vertices and hyper-edges. While an edge connects two vertices of a graph, a hyper-edge can connect any number of vertices. For each clustering vector $\lambda^{(q)}$ a binary indicator matrix $H^{(q)}$ can be defined with n rows and $k^{(q)}$ columns. $H_{i,j}^{(q)}$ is 1 if x_i was placed in cluster j in clustering $\lambda^{(q)}$. The entire ensemble of clusterings can hence be represented by a concatenation of individual indicator matrices as $H = (H^{(1)}, ..., H^{(r)})$. The matrix H, now, defines a hyper-graph with n vertices and $\sum_{q=1}^{r} k^{(q)}$ hyper-edges. Each hyper-edge connects all the vertices that have a value 1 in the corresponding column. This transformation of $\lambda^{(1,2,\ldots,r)}$ to H is shown in Tables 1.1 and 1.2.

1.2.2 Graph-Theoretic Approaches

Upon formulating the cluster ensemble problem, Strehl and Ghosh (2002) proposed three graph-theoretic approaches (CSPA, HGPA, and MCLA) for finding the consensus clustering. Later Fern and Brodley (2004) proposed the HBGF algorithm that is based on bipartite graph partitioning. All these approaches use the efficient graph partitioning algorithm METIS by Karypis and Kumar (1998) to partition graphs induced by the cluster ensemble and find the consensus clustering. Note that there is implicitly an additional constraint in these solutions, namely that the consensus clusters obtained be of comparable size. We describe these and other algorithms in the following subsections.

1.2.2.1 Cluster-based Similarity Partitioning Algorithm (CSPA)

In CSPA the similarity between two data-points is defined to be directly proportional to number of constituent clusterings of the ensemble in which they are clustered together. The intuition is that the more similar two data-points are the higher is the chance that constituent clusterings will place them in the same cluster. Hence, in this approach a $n \times n$ similarity matrix is computed as $W = \frac{1}{r}HH^T$. This similarity matrix (graph) can be clustered using any reasonable pair wise similarity based clustering algorithm to obtain the final clustering. In CSPA the authors chose METIS to partition the similarity graph to obtain the desired number of clusters. Because CSPA constructs a fully connected graph its computational and storage complexity are $\mathcal{O}(n^2)$. Hence, it is more expensive in terms of resources than algorithms that will be introduced next.

1.2.2.2 Hyper-Graph Partitioning Algorithm (HGPA)

The HGPA algorithm seeks to directly partition the hyper-graph defined by the matrix H in Table 1.2. Hyper-graph partitioning seeks a to cluster the data by eliminating the minimal number of hyper-edges. This partitioning is performed by the package HMETIS by Karypis et al. (1997). In the HGPA algorithm all the vertices and hyper-edges are weighted equally. In our experiments, HGPA displayed a lack of robustness and routinely performed worse than the CSPA and MCLA algorithms. Hence, we will not discuss this algorithm or report any results for it in the remainder of this chapter.

1.2.2.3 Meta-CLustering Algorithm(MCLA)

The MCLA algorithm takes a slightly different approach to finding the consensus clustering than the previous two methods. First, it tries to solve the cluster correspondence problem and then uses voting to place data-points into the final consensus clusters. The cluster correspondence problem is solved by grouping the clusters identified in the individual clusterings of the ensemble.

As we have seen earlier, the matrix H represents each cluster as n-length binary vectors. In MCLA, the similarity of cluster c_i and c_j is computed based on the number of data-points that are clustered into both of them, using the Jaccard measure $W_{i,j} = \frac{|c_i \cap c_j|}{|c_i \cup c_j|}$. This similarity matrix (graph), with clusters as nodes, is partitioned into *meta-clusters* using METIS.

The final clustering of instances is produced in the following fashion. All the clusters in each meta-cluster are collapsed to yield a association vector for the meta-cluster. This association vector for a meta-cluster is computed by averaging the association of instances to each of the constituent clusters of that meta-cluster. The instance is then clustered into the meta-cluster that it is most associated to.

The cluster similarity matrix can be computed in time quadratic in the number of clusters in the ensemble. This is often significantly less than n^2. Furthermore, the averaging and voting operations are linear in n. This makes MCLA computationally very efficient.

1.2.2.4 Hybrid Bipartite Graph Formulation (HBGF)

This method was introduced by Fern and Brodley (2004) with an aim to model the instances and clusters simultaneously in a graph. The CSPA algorithm models the ensemble as a graph with the vertices representing instances in the data, while the MCLA algorithm models the ensemble as a graph of clusters. The HBGF technique combines these two ideas and represents the ensemble by a bipartite graph in which the individual data points and the clusters of the constituent clusterings are both vertices. The graph is bipartite because there are no edges between vertices that are both either instances or clusters. The complete set of rules to assign the weights on the edges is as follows:

- $W(i,j) = 0$ if i, j are both clusters or both instances
- $W(i,j) = 0$ if instance i doesn't belong to cluster j
- $W(i,j) = 1$ if instance i belongs to cluster j

This bipartite graph is partitioned into k parts yielding the consensus clustering. The clustering is performed using METIS and Spectral clustering (Ng et al. 2001). The clusters in the consensus clustering contain both instances and the original clusters. Hence, the method yields a co-clustering solution. This method has also been previously used to simultaneously cluster words and documents by Dhillon (2001).

The computational complexity of HBGF is $\mathcal{O}(n \times t)$, where t is the total number of clusters in the ensemble. While this is significantly less than quadratic in the number of instances (as in CSPA), in practice we observe the algorithm to be fairly resource hungry both in terms of CPU time and storage.

1.2.2.5 Evidence Accumulation Framework

Evidence Accumulation (Fred and Jain 2001, 2002) is a simple framework, very similar to the cluster ensemble framework, for combining the results of multiple weak clusterings in order to increase robustness of the final solution. The framework uses a K-Means type algorithm to produce several clusterings each with a random initialization. The number of clusters specified in each KMeans clustering is typically much larger than the true number of clusters desired. The data instances are then mapped into the similarity space where the similarity between two instances i and j is the fraction of clusterings in which they ended up in the same cluster. A Minimum Spanning-Tree based clustering algorithm is then used to obtain the final clustering. In practice any appropriate clustering technique could be employed. This framework and the consensus function that it uses are very similar to the Cluster Ensemble framework and the CSPA algorithm (Strehl and Ghosh 2002).

A similar framework for obtaining robust clustering solutions has been proposed by Frossyniotis et al. (2002). The actual consensus function used in this algorithm only works on heavily restricted type of ensembles; each constituent clustering has the same number of clusters. Also, Fern and Brodley (2003) extended this approach to accept soft clusterings as input. The details of this approach are presented in Subsection 1.3.4.

1.2.3 Ensemble as a Mixture of Multinomials

Topchy et al. (2004) model the ensemble, $\lambda^{(1,2,\ldots,r)}$, using a generative model and use EM to estimate the parameters of the model. The EM procedure along with the parameters provides us with a soft final clustering.

In this approach, it is assumed that the ensemble has been generated from a mixture of multi-dimensional multinomial distributions. Each data point is generated by first picking a multinomial distribution according to the priors. After picking a component of the mixture the cluster label in each clustering is picked from a multinomial distribution over the cluster labels. The cluster labels of different constituent clusterings are assumed to be i.i.d..

The number of parameters to be estimated increases with both the number of constituent clusterings as well as with the number of clusters in them. Experiments in Topchy et al. (2004) do not include experiments on datasets that have more than 3 clusters. In this chapter we will evaluate the performance of this consensus function on more complex real-life datasets.

One advantage of this approach is that it is easy to model final clusters of different sizes using this method. Graph partitioning methods tend to yield roughly balanced clusters. This is a disadvantage in situations where the data distribution is not uniform. Using the priors in the mixture model the distribution of data can be accommodated conveniently.

1.2.4 Diversity in Cluster Ensembles

Diversity among the classifiers in an ensemble has been shown to improve its accuracy (Hansen and Salamon 1990; Melville and Mooney 2003). Here, we recount some research on the impact of diversity on cluster ensembles.

Ghosh et al. (2002) examine the problem of combining multiple clusters of varying resolution and showed that it is possible to obtain robust consensus even when the number of clusters in each of the individual clusterings is different. They also describe a simple scheme for selecting a “good” number of clusters k for the consensus clustering by observing the variation in average normalized mutual information with different k. Fern and Brodley (2003) report on some experiments on diversity of ensembles. They find that the consensus function’s accuracy increases as the ensemble is made more diverse. Kuncheva and Hadjitodorov (2004) study the diversity of ensembles using multiple measures like the Rand Index, Jaccard measure etc. Based on this study they propose a variant of the Evidence Accumulation framework where the number of over-produced clusters is randomly chosen. This randomization in ensemble generation is shown to increase the diversity of the ensembles thereby leading to better consensus clustering. In a recent follow up work Hadjitodorov et al. (2006) report that selecting constituent clusterings based on median diversity leads to better ensembles.

1.3 Soft Cluster Ensembles

In this section we will formally define the soft cluster ensemble problem and provide intuition on why we expect soft cluster ensembles to yield better results than their corresponding hard versions. We will then introduce a new algorithm based on Information Theoretic KMeans (Dhillon et al. 2003b) to solve ensembles of soft clusterings. In order to objectively evaluate our new approach, will describe changes to existing techniques mentioned in Section 1.2 to enable them to handle soft ensembles.

Table 1.3 A set of three clusterings

	$\lambda^{(1)}$	$\lambda^{(2)}$	$\lambda^{(3)}$
x_1	1	2	1
x_2	1	2	1
x_3	1	3	2
x_4	2	3	2
x_5	2	3	3
x_6	3	1	3
x_7	3	1	3

Table 1.4 Ensemble of soft clusterings

	$S^{(1)}$			$S^{(2)}$			$S^{(3)}$		
	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9
x_1	0.7	0.2	0.1	0.1	0.7	0.2	0.6	0.3	0.1
x_2	0.9	0.1	0.0	0.0	0.8	0.2	0.8	0.2	0.0
x_3	0.9	0.0	0.1	0.1	0.4	0.5	0.5	0.5	0.0
x_4	0.2	0.6	0.2	0.1	0.2	0.7	0.2	0.7	0.1
x_5	0.1	0.9	0.0	0.0	0.1	0.9	0.0	0.5	0.5
x_6	0.0	0.2	0.8	0.8	0.1	0.1	0.1	0.2	0.7
x_7	0.1	0.2	0.7	0.7	0.1	0.2	0.1	0.3	0.6

1.3.1 The Soft Cluster Ensemble Problem

In order to facilitate the explanation of various algorithms later in this section we now define the soft cluster ensemble problem formally.

As in the case of hard ensembles, let $X = \{x_1, x_2, ..., x_n\}$ denote a set of instances/objects. Also, let $\lambda^{(q)} \in \{1, 2, ...k^{(q)}\}^n$ denote the label vector of the q^{th} clustering of X; *i.e.* $\lambda_i^{(q)}$ is the label of x_i in the q^{th} clustering. This is the hard labeling defined in Subsection 1.2.1. In cases where the underlying clustering algorithm outputs soft cluster labels, $\lambda_i^{(q)}$ is defined as $argmax_j P(C_j|x_i)$, where $P(C_j|x_i)$ is the posterior probability of instance x_i belonging to cluster C_j. A soft cluster ensemble is shown in Table 1.4 and its corresponding hard version in Table 1.3.

Instead of *hardening* the posterior probabilities into cluster labels we construct a matrix $S^{(q)}$ representing the solution of the q^{th} soft clustering algorithm. $S^{(q)}$ has a column for each cluster generated in the clustering and the rows denote the instances of data with $S_{ij}^{(q)}$ being the probability of x_i belonging to cluster j of the q^{th} clustering. Hence, the values in each row of $S^{(q)}$ sum up to 1. There are r such clusterings ($S^{(1,...,r)}$) each with $k^{(q)}$ clusters. Just as in the hard ensemble problem, our goal is to find a consensus function Γ which combines these clusterings into a combined labeling, λ, of the data. It should be noted that the cluster ensemble framework doesn't specify whether the final clusterings should be hard or soft. In this chapter we only work with algorithms that output hard final clusterings.

1.3.2 Intuition behind Soft Ensembles

It is fairly obvious from the above definition that hardening a soft cluster ensemble entails a loss of information. But, it is not at all obvious that this additional information is useful. The goal of this study is to show empirically that algorithms designed for soft ensembles improve upon the accuracy of those that operate on the hardened versions of the ensembles. Here, we will try to intuitively explain why we expect this.

For the sake of discussion consider a cluster ensemble where individual clusterings are working on vertically partitioned data. In such a scenario, the underlying clustering algorithms have access to different and often incomplete sets of features. Incomplete data could result from distributed computing constraints (Ghosh et al. 2002), random projections in order to facilitate high dimensional clustering (Fern and Brodley 2003), or multi-view datasets as used in (Kreiger and Green 1999). Under such circumstances there is an increased chance that the underlying clustering algorithms will not be able to assign some objects into clusters with much certainty. If the combining procedure were to accept only hard clusterings, these objects would have to be assigned to the cluster they most belong to (one with the highest posterior probability).

Consider the soft ensemble depicted in Table 1.4. The solution $S^{(2)}$ assigns x_3 to clusters s_4,s_5, and s_6 with probabilities 0.1, 0.4, and 0.5 respectively. If the consensus function were to only accept hard clusterings it would be provided with a vector where $\lambda_i^{(2)}$ is s_6. The combining algorithm would have no evidence that the 2^{nd} underlying clustering algorithm was unsure about the assignment of x_3. It would accept this observation with the same amount of certainty as any other observations that assigns a data-point x_i to a cluster s_j with 0.9 probability. If, however, the combining function were to accept soft clusterings, it could potentially use this information to make appropriate cluster assignment of x_3 in the combined clustering. Since it's more likely that clustering algorithms are unsure of their assignments while operating with incomplete set of features, it is important that the combining function have access the cluster assignment probabilities, and not just the hard assignments themselves.

1.3.3 Solving Soft Ensembles with Information-Theoretic KMeans (ITK)

Information-Theoretic KMeans was introduced by Dhillon et al. (2003b) as way to cluster words in order to reduce dimensionality in the document clustering problem. This algorithm is very similar to the KMeans algorithm, differing only in the fact that as a measure of distance it uses the KL-divergence (Kullback and Leibler 1951) instead of the Euclidean distance. The reader is referred to the original paper for more details. Here we just describe the mapping of the soft cluster ensemble problem to the information-theoretic K-Means problem.

Each instance in a soft ensemble is represented by a concatenation of r posterior membership probability distributions obtained from the constituent clustering algorithms (see matrix S in Table 1.4). Hence, we can define a distance measure between two instances using the Kullback-Leibler (KL) divergence (Kullback and Leibler 1951), which calculates the "distance" between two probability distributions. The distance between two instances can be calculated as

$$KL_{v_a,v_b} = \sum_{q=1}^{r} w^{(q)} \sum_{i=1}^{k^{(q)}} S^{(q)}_{v_a i} log \left(\frac{S^{(q)}_{v_a i}}{S^{(q)}_{v_b i}} \right) \tag{1.1}$$

where, $w^{(q)}$ are clustering specific weights, such that $\sum_{q=1}^{r} w^{(q)} = 1$.

Equation (1.1) computes pairwise distance by taking an average of the KL divergence between the two instances in individual constituent clusterings. Here we note that this is equivalent to computing the KL divergence between instances represented by a matrix S in which each row sums upto one. This normalization can be performed by multiplying each value in $S^{(q)}$ by $\frac{w^{(q)}}{\sum_{q=1}^{r} w^{(q)}}$. Now that we have a distance measure between instances based on KL-divergence, we can use existing information-theoretic K-Means software mentioned above to solve the soft ensemble.

Computing Equation (1.1) with $w^{(q)} = \frac{1}{r}$ assumes that all the clusterings are equally important. We can, however, imagine a scenario where we have different importance values for the constituent clusterings. These values could, for instance, be our confidence in the accuracy of these clusterings, possibly based on the number of features they access. These confidence values can be easily integrated into the cost function using the weights $w^{(q)}$.

1.3.4 Soft version of CSPA (sCSPA)

The CSPA algorithm proposed by Strehl and Ghosh (2002) works by first creating a co-association matrix of all objects, and then using METIS (Karypis and Kumar 1998) to partition this similarity space to produce the desired number of clusters. This algorithm is described in Section 1.2.

sCSPA extends CSPA by using values in S to calculate the similarity matrix. If we visualize each object as a point in $\sum_{q=1}^{r} k^{(q)}$ dimensional space, with each dimension corresponding to probability of its belonging to a cluster, then SS^T is the same as finding the dot product in this new space. Thus the technique first transforms the objects into a *label-space* and then interprets the dot product between the vectors representing the objects as their similarity. In our experiments we use Euclidean distance in the label space to obtain our similarity measure. The dot product is highly co-related with the Euclidean measure, but Euclidean distance provides for cleaner semantics. Euclidean distance between v_a and v_b is calculated as

$$d_{v_a,v_b} = \sqrt{\sum_{q=1}^{r} \sum_{i=1}^{k^{(q)}} \left(S^{(q)}_{v_a i} - S^{(q)}_{v_b i} \right)^2}$$

This can be interpreted as a measure of the difference in the membership of the objects for each cluster. This dissimilarity metric is converted into a similarity measure using $s_{v_a,v_b} = e^{-d^2_{v_a,v_b}}$.

Another distance measure can be defined on the instances in a soft ensemble using KL-divergence (Kullback and Leibler 1951) as in Section 1.3.3. In our results we observed that all versions of the sCSPA (with Euclidean distance, KL divergence, and cosine similarity) gave very similar results. The results obtained while using Euclidean distance were sometimes better, so here we will report results based on only that version of the sCSPA. sCSPA (like

CSPA) is impractical for large datasets, and hence we will only report results for datasets with less than 2000 data-points.

Fern and Brodley (2003) proposed a variant of the Evidence Accumulation framework that accepts soft clusterings. In this scenario, the similarity of two instances is calculated as the average dot product of the probability distributions describing them. Hence,

$$sim(v_a, v_b) = \frac{1}{r} \sum_{i=1}^{k^{(q)}} S_{v_a i}^{(q)} \times S_{v_b i}^{(q)}$$

The similarity matrix that results is then clustered using a complete-link agglomerative algorithm. The input matrix used by this framework is essentially equivalent to the one used by sCSPA (using Cosine similarity). The only difference is in the combining function. Hence, we will not experiment with this technique further in this chapter.

1.3.5 Soft version of MCLA (sMCLA)

In MCLA each cluster is represented by a n-length binary association vector. The idea is to group and collapse related clusters into meta-clusters, and then assign each object to the meta-cluster in which it belongs most strongly. The clusters are grouped by graph partitioning based clustering.

sMCLA extends MCLA by accepting soft clusterings as input. sMCLA's working can be divided into the following steps (similar steps are followed in MCLA too).

Construct Soft Meta-Graph of Clusters: All the $\sum_{q=1}^{r} k^{(q)}$ clusters or indicator vectors s_j (with weights), the hyper-edges of S, can be viewed as vertices of another regular undirected graph. The edge weights between two clusters s_a and s_b is set as $W_{a,b} = Euclidean\_dist(s_a, s_b)$. The Euclidean distance is a measure of the difference of membership of all objects to these two clusters. As in the sCSPA algorithm, the Euclidean distance is converted into a similarity value.

Group the Clusters into Meta-Clusters: The meta-graph constructed in the previous step is partitioned using METIS to produce k balanced meta-clusters. Since each vertex in the meta-graph represents a distinct cluster label, a meta-cluster represents a group of corresponding cluster labels.

Collapse Meta-Clusters using Weighting: We now collapse all the clusters contained in each meta-cluster to form its association vector. Each meta-cluster's association vector contains a value for every object's association to it. This association vector is computed as the mean of the association vectors for each cluster that is grouped into the meta-cluster. This is a weighted form of the step performed in MCLA.

Compete for Objects: Each object is assigned to the meta-cluster to which it is most associated. This can potentially lead to a soft final clustering, since the ratio of the winning meta-cluster's association value to the sum of association values of all final meta-clusters can be the confidence of assignment of an object to the meta-cluster.

There is, however, one problem with this approach. Because we are using soft clusterings as inputs, the co-association graph of the clusters (meta-graph) is almost complete. More specifically, even clusters from the same clusterings have non-zero similarity to each other. This is not the case with MCLA since it uses a binary Jaccard measure, and for hard clusterings Jaccard similarity between clusters in the same clusterings is necessarily zero. We get better consensus clustering results after making the co-association matrix r-partite. Hence, sMCLA forces the similarity of hyper-edges coming from the same clustering to be zero. This is, however, only done when the number of clusters in all the constituent clusterings is equal to the desired final number of clusters. In ensembles where the number of clusters in each underlying clustering vary the algorithm does not force the co-association matrix to be r-partite.

1.3.6 Soft version of HBGF (sHBGF)

HBGF represents the ensemble as a bipartite graph with clusters and instances as nodes, and edges between the instances and the clusters they belong to. This approach can be trivially adapted to consider soft ensembles since the graph partitioning algorithm METIS accepts weights on the edges of the graph to be partitioned. In sHBGF, the graph has $n + t$ vertices, where t is the total number of underlying clusters. The weights on the edges are set as follows:

- $W(i, j) = 0$ if i, j are both clusters or both instances
- $W_{(i,j)} = S_{i,j}$ otherwise, where i is the instance and j is the cluster

1.4 Experimental Setup

We empirically evaluate the various algorithms presented in Sections 1.2 and 1.3 on soft cluster ensembles generated from various datasets. In this section we describe the experimental setup in detail.

1.4.1 Datasets Used

We perform the experimental analysis using the six real-life datasets and one artificial dataset. Some basic properties of these datasets are summarized in Table 1.5. These datasets were selected so as to present our algorithms with problems of varying degrees of difficulty – in terms of number of desired clusters, number of attributes, and number of instances. All these datasets, with the exception of 8D5K and HyperSpectral, are publicly accessible from the UCI data repository (Blake and Merz 1998).

- **8D5K**: This is an artificially generated dataset containing 1000 points. It was generated from 5 multivariate Gaussian distributions (200 points each) in 8-dimensional space. The clusters all have the same variance but different means. The means were drawn from a uniform distribution within the unit hypercube. This dataset was used in (Strehl and Ghosh 2002) and can be obtained from `http://www.strehl.com`.
- **Vowel**: This dataset contains data on the pronunciation of vowels. We removed some nominal features which corresponded to the context like sex, name etc, and only retained the 10 real valued features. There are 11 classes in the data and an average of 93 instances per class.
- **Pendigits**: This dataset was generated for the problem of pen-based recognition of handwritten digits. It contains 16 spatial features for each of the 10992 instances. There

Table 1.5 Datasets used in experiments

Name	Type of features	#features	#classes	#instances
8D5K	real	8	5	1000
Vowel	real	10	11	990
Pendigits	real	16	10	10992
Glass	real	9	6	214
HyperSpectral	real	30	13	5211
Yeast	real	8	10	1484
Vehicle	real	18	4	846

are 10 classes of roughly equal sizes corresponding to the digits 0 to 9. In order to get better clustering results, we normalized the columns (features) to sum to 1.

- **Glass**: The instances in this dataset are samples of glass used for different purposes. Real-valued features corresponding to their chemical and optical properties describe the instances. There are 214 instances categorized into 6 classes such as tableware, containers etc based on 9 attributes.

- **HyperSpectral**: This dataset contains 5211 labeled pixel from a HyperSpectral snapshot of the Kennedy Space Center. Each data-point is described by a set of 30 HyperSpectral signatures pruned from an initial set of 176 features. The pruning was performed by a best-basis feature extraction procedure (Kumar et al. 2001). The dataset has 13 classes describing the geographical features apparent in the pixel.

- **Yeast**: The Yeast dataset contains information about proteins within Yeast cells with the class attribute denoting the localization within the cell. This is a fairly hard problem, and this shows in the clustering results we obtain. The 1484 instances are each characterized by 8 attributes, and there are 10 classes in the dataset.

- **Vehicle**: This dataset was designed for the purpose of learning to classify a given silhouette as one of four types of vehicle, using a set of 18 features extracted from the silhouette. The vehicle may be viewed from one of many different angles. The 846 silhouette instances are classified into 4 vehicle categories Opel, Saab, Bus, and Van.

1.4.2 Ensemble Test-set Creation

In order to compare the hard and soft ensemble methods, as well as to evaluate the our Information-Theoretic KMeans (ITK) based approach, we created soft cluster ensembles of varying degrees of difficulty. Note here that for each soft cluster ensemble we also stored its corresponding hardened version to evaluate methods that only accept hard clusterings.

The individual clusterings in our ensembles were created using the EM algorithm (Dempster et al. 1977) with a mixture of Gaussian distributions model, but any algorithm that outputs soft probabilities could have been used. Further, each constituent clustering was created using

Table 1.6 Dataset specific options for creating ensembles

Name	# attributes	Numatts options	#clusterings/Numatts-option
8D5K	8	3,4,5,6	10
Vowel	10	3,4,5,6,7	10
Pendigits	16	3,4,6,9,12	15
Glass	9	3,4,5,6,7	10
HyperSpectral	30	5,10,15,20,25	15
Yeast	8	2,3,4,5	10
Vehicle	18	4,5,8,11	15

vertically partitioned subsets of the datasets. This partial view of the data as well as the dependence of EM on initialization resulted in the diversity in the individual clustering solutions in an ensemble.

As mentioned above, we wanted to evaluate our algorithms on ensembles of varying degrees of difficulty. For this purpose we created ensembles by varying two parameters that controlled the degree of difficulty. The first parameter is the number of attributes that the EM algorithm accesses while creating the constituent clusterings. We expect the difficulty of an ensemble containing clusterings created from less attributes to be higher. The second parameter is the number of constituent clusterings in the ensemble. In general, we expect that as the number of constituent clusterings increase the consensus clusterings obtained should be more accurate. For most datasets the number of clusterings in the ensembles is varied from 2 to 10, and in some cases to 15. The entire set of options for all the datasets is listed in Table 1.6. The second column in the table describes the different settings for number of features used to create clusterings. For instance, for the 8D5K dataset we can obtain ensembles with constituent clusterings created using 3,4,5, or 6 attributes. Also, for each such setting we can select from 10 clusterings to form an ensemble. Of course, each of these 10 clusterings is created with a randomly selected set of attributes.

Hence, while creating an ensemble we specify three parameters: the dataset name, the number of attributes, and the number of clusterings. For each set of parameter values, we create multiple ensembles by randomly selecting the clusterings to combine. Also, non-deterministic consensus functions are run multiple times in order to average out variations in results due to initialization.

Here we must note that each individual clustering as well as the consensus function is given the true number of clusters to find. The use of ensembles for finding the true number of clusters, or the effect of different k in constituent clusterings on ensemble accuracy are not investigated in this study.

1.4.3 Evaluation Criteria

In order to evaluate the final consensus clusterings obtained we use two different criteria. Both these criteria compare the obtained clustering to the true labels of the instances. We also use the Geometric Mean Ratio to present an overall score for the performance of each algorithm.

1.4.3.1 Normalized Mutual Information (NMI)

The first criterion we use was introduced by Strehl and Ghosh (2002). and is called Normalized Mutual Information (NMI).

The NMI of two labellings of instances can be measured as

$$NMI(X,Y) = \frac{I(X,Y)}{\sqrt{H(X)H(Y)}} \tag{1.2}$$

where, $I(X,Y)$ denotes the mutual information between two random variables X and Y and $H(X)$ denotes the entropy of X. In our evaluation, X will be consensus clustering while Y will be the true labels.

NMI has some nice properties such as $NMI(X,X) = 1$ and if Y has only one cluster label for all instances $NMI(X,Y) = 0$. With these properties NMI is extensively used for evaluating clustering algorithms in literature.

Another measure of clustering accuracy is Adjusted RAND (Hubert and Arabie 1985). The Adjusted RAND compares two labellings based on whether pairs of objects are placed in the same or different clusters in them. The maximum value it takes is 1, and its expected value is 0. We computed the Adjusted RAND score for each solution and found it to be highly correlated to the NMI score. Hence we will only report the NMI score in the chapter.

1.4.3.2 Classification via Clustering (CVC)

The CVC is a measure of the purity of the clusters obtained w.r.t. the ground truth. The CVC is calculated by the following procedure.

- To each cluster, assign the label that corresponds to a majority of points.
- Each instance is now labeled by its cluster's label.
- CVC is the fraction of misclassified instances in such a classification of instances.

The CVC measure weighs the contribution of a cluster to the average by its size. This ensures that very small pure clusters don't compensate for large impure ones.

There are other issues with this measure, however. The CVC measure is biased towards solutions with large number of very small pure clusters. This is not an issue in our evaluation since the number of output clusters is kept constant across all the consensus functions being compared. Also, the CVC measure is not very well defined in case of empty clusters in the clustering solution. Since we ignore the purity of empty clusters in our calculation of CVC, if all the instances were clustered into one cluster, CVC would be the fraction of instances that belong to the class with the largest number of instances. NMI would have been zero in such a case. This is not a problem for most datasets since many algorithms are based on graph partitioning approaches and output balanced clusters. But like most existing literature on cluster ensembles, we will use NMI as our principal measure of goodness.

1.4.3.3 Geometric Mean Ratio

Since we are varying the ensemble parameters over a very wide range for each dataset, we end up with a lot of different points of comparison. In order to report some sort of overall score for each algorithm on all the ensembles used, we use the Geometric Mean Ratio (Webb 2000). The GMR is calculated as follows. Suppose we have n ensembles that we tested our algorithms on, and NMI_A and NMI_B are vectors of the average NMI values w.r.t. to true labels obtained by algorithms A and B on these runs. GMR is calculated as

$$GMR(A, B) = \left(\prod_{i=1}^{n} \frac{NMI_{Bi}}{NMI_{Ai}} \right)^{\frac{1}{n}} \tag{1.3}$$

In later sections we display the GMR values in tables with rows and columns representing the algorithms being compared. In these tables element (i, j) represents the value $GMR(algo(i), algo(j))$, where $algo(i)$ and $algo(j)$ are the algorithms represented in row i and column j respectively. Hence, values > 1 along a column mean that the algorithm corresponding to the column performs better than the other algorithms. Similarly, the values < 1 along the rows indicates that the algorithm corresponding to the row scores better than the other algorithms.

1.5 Soft vs Hard Cluster Ensembles

In this section we present results from our evaluation of the algorithms we described in earlier sections using the experimental setup described Section 1.4. In Subsection 1.5.1 we will compare the performance of algorithms accepting soft ensembles as input and those that run on hardened versions of the ensembles. After analyzing these experiments we will compare the Information-Theoretic KMeans (ITK) approach with the best performing algorithms from Subsection 1.5.1. Finally, in Subsection 1.5.3 and Subsection 1.5.4, we will examine the variation in performance of algorithms on ensembles of varying difficulty.

1.5.1 Soft Versions of Existing Algorithms

In this section we evaluate the performance of CSPA, MCLA, and HBGF, their soft counterparts, and the Mixture of Multinomials method. The evaluation measure we employ is the Geometric Mean Ratio (GMR), which is calculated over all the ensembles that were created as described in Subsection 1.4.2. There were, however, some exceptions to the direct application of the GMR formula over all datasets. HBGF, CSPA and their soft versions were not run on the HyperSpectral and Pendigits datasets because these datasets are too large to expect solutions in reasonable time. Hence, when we compare one of these algorithms to the others we do not consider ensembles of these large datasets. Also, in certain cases (for hard ensembles) the consensus functions output clusterings that score 0 on the NMI measure. This would happen, for example, if all the instances were placed in a single cluster. In such cases the GMR either becomes 0 or ∞ depending on where the zero score appears. Hence, we assign a very small nominal value (0.00001) to the NMI score whenever it is zero. The effect of this nominal score vanishes because we normalize by taking the n^{th} root of the product.

Table 1.7 shows the GMR values of the NMI measure comparing the three original algorithms as well as their soft versions. We can see that for each algorithm the soft version performs better than the corresponding hard version. Keep in mind that algorithm with values < 1 on the rows are performing better than the others. The table shows that averaged over all the ensembles we created, the soft versions of the algorithms are slightly better than their

Table 1.7 Geometric mean ratio of NMI score over all ensembles. The value $table_{i,j}$ indicates ratio of algorithms j/i

Dataset	CSPA	sCSPA	MCLA	sMCLA	HBGF	sHBGF	MixMns
CSPA	1	1.05	0.718	0.999	0.978	1.02	0.802
sCSPA	0.94	1	0.68	0.948	0.928	0.967	0.76
MCLA	1.163	1.22	1	1.17	1.136	1.18	0.913
sMCLA	1.00	1.05	0.56	1	0.978	1.019	0.77
HBGF	1.02	1.076	0.73	1.02	1	1.04	0.82
sHBGF	0.98	1.03	0.705	0.98	0.959	1	0.787
MixMns	1.25	1.31	0.73	1.297	1.219	1.269	1

Table 1.8 Geometric mean ratio of CVC score over all ensembles. The value $table_{i,j}$ indicates ratio of algorithms j/i

Dataset	CSPA	sCSPA	MCLA	sMCLA	HBGF	sHBGF	MixMns
CSPA	1	1.02	0.795	1.17	0.99	1.01	0.964
sCSPA	0.976	1	0.777	1.146	0.97	0.99	0.94
MCLA	1.015	1.039	1	1.197	1.01	1.03	0.99
sMCLA	0.85	0.873	0.53	1	0.85	0.87	0.80
HBGF	1.004	1.029	0.799	1.179	1	1.02	0.97
sHBGF	0.98	1.009	0.78	1.156	0.98	1	0.95
MixMns	1.037	1.06	0.66	1.24	1.03	1.05	1

hard counterparts. This shows that the soft versions of the algorithms are able to use the extra information in the soft ensembles to obtain better consensus clusterings.

We notice that the mixture of Multinomials algorithm (MixMns) performs worse than all other algorithms other than MCLA. This may be because many of the datasets we used had a large number of clusters, causing parameter estimation problems for the mixture model. Topchy et al. (2004) only evaluated their algorithm on real datasets with very low number of clusters.

Another key observation is the dramatic difference in the performance of the sMCLA and MCLA algorithms. The performance improvement of sMCLA over MCLA is by far larger than the improvements by other soft versions like sCSPA and sHBGF. This is because MCLA's performance is very bad when the input clusterings are not accurate. This can be seen by its performance values over tough ensembles (Table 1.9) as well as ensembles with very low number of attributes in constituent clusterings (Figure 1.1). sMCLA doesn't get misled during the meta-clustering phase because the distances between the clusters are now determined from soft probabilities. Hence, an error in a input clustering which assigns an instance into the wrong cluster could be alleviated in sMCLA's case if the posterior probabilities of the wrong assignment are small. This phenomenon, however, needs to be investigated further since sMCLA performs on par with the best algorithms shown in Table 1.7.

Table 1.9 Geometric mean ratio of NMI score over tough ensembles. The value $table_{i,j}$ indicates ratio of algorithms j/i

Dataset	CSPA	sCSPA	MCLA	sMCLA	HBGF	sHBGF	MixMns
CSPA	1	1.085	0.652	0.997	0.97	1.06	0.655
sCSPA	0.92	1	0.60	0.919	0.897	0.98	0.604
MCLA	1.53	1.665	1	1.47	1.49	1.63	0.922
sMCLA	1.003	1.088	0.46	1	0.976	1.06	0.627
HBGF	1.028	1.113	0.67	1.025	1	1.09	0.673
sHBGF	0.94	1.024	0.62	0.94	0.92	1	0.618
MixMns	1.53	1.656	0.73	1.59	1.485	1.617	1

Table 1.8 shows the GMR value table for the CVC measure. As we can see from the table the GMR values closely correspond to the values in the Table 1.7. Since the values in the two tables closely agree we will henceforth only report results using the NMI measure.

In order to evaluate the intuition that the information obtained from soft ensembles is especially useful when dealing with *tough* ensembles, we have populated the Table 1.9 with GMR values calculated over only the *tough* ensembles. *Tough* ensembles are defined as those comprising a small number of clusterings, each of which are obtained using very few features. In our experiments, tough ensembles contained only 2-4 clusterings which were obtained using the minimum Numatts option number of features for each dataset shown in Table 1.6. For example, a tough ensembles for the 8D5K dataset might contain 3 clusterings, each obtained using only 3 features. As we can see from Table 1.9, soft versions of algorithms perform better than their hard counterparts and the difference in their performance is slightly higher than those in Table 1.7. The fact that the differences in performances are higher shows that the extra information in soft clusterings is useful in tough situations.

1.5.2 Information-Theoretic KMeans (ITK)

We compare the Information-Theoretic KMeans algorithm with only two of the best algorithms from the analysis in the previous section. Table 1.10 displays the GMR values for the ITK, sHBGF, and sMCLA algorithm over all the ensembles. As we can see the ITK algorithm performs appreciably better than both sHBGF and sMCLA. The sHBGF and sMCLA algorithm are fairly similar to each other in overall performance. The Geometric mean ratio matrix for the CVC score is identical to the one for the NMI score, and we don't report those results.

In order to find whether ITK performs better for tougher or simpler ensembles we calculate GMR over only the tough ensembles. Here again the tough ensembles are defined as in Subsection 1.5.1. The results of this experiment are listed in Table 1.11. As we can see from the two tables the improvement in ITK algorithm's performance over sHBGF/sMCLA is higher for the subset of tougher ensembles.

In the set of datasets selected for this chapter some present tougher challenges to the clustering algorithms than others. In terms of the NMI score of clusterings 8D5K is the simplest dataset while Yeast is the toughest. We display in Table 1.12 and Table 1.13 the

Table 1.10 Geometric mean ratio of NMI score over all ensembles. The value $table_{i,j}$ indicates ratio of algorithms j/i

Dataset	ITK 10K	sHBGF	sMCLA
ITK 10K	1	0.856	0.875
sHBGF	1.167	1	0.98
sMCLA	1.142	1.012	1

Table 1.11 Geometric mean ratio of NMI score over tough ensembles. The value $table_{i,j}$ indicates ratio of algorithms j/i

Dataset	ITK 10K	sHBGF	sMCLA
ITK 10K	1	0.816	0.798
sHBGF	1.226	1	0.94
sMCLA	1.253	1.06	1

Table 1.12 Geometric mean ratio of NMI score for only the 8d5k dataset. The value $table_{i,j}$ indicates ratio of algorithms j/i

Dataset	ITK 10K	sHBGF	sMCLA
ITK 10K	1	1.03	0.97
sHBGF	0.968	1	0.944
sMCLA	1.025	1.05	1

GMR value matrix for ensembles of datasets 8D5K and Yeast respectively. As we can see from these tables, in the case of the Yeast dataset ITK is by far the best performing algorithm. But for the 8D5K dataset all algorithms are fairly comparable with sHBGF slightly better than the rest. One reason is that for soft ensembles where most probability values are close to 1 or 0, more complex algorithms like ITK do not perform better than simple graph-theoretic approaches.

Another explanation for ITK's performance on the Yeast dataset can be provided based on the characteristics of the algorithms. The graph partitioning based consensus algorithms are constrained to provide roughly balanced clusters. This can be a problem in cases where the underlying data does not have balanced classes. The 8D5K dataset has perfectly balanced clusters (200 instances each) while the Yeast dataset has classes that range from 5 instances to 463 instances in size. The ITK algorithm is not constrained to find balanced clusters and hence can adapt the clustering solution better to the natural distribution of instances in the

Table 1.13 Geometric mean ratio of NMI score for only the yeast dataset. The value $table_{i,j}$ indicates ratio of algorithms j/i

Dataset	ITK 10K	sHBGF	sMCLA
ITK 10K	1	0.84	0.68
sHBGF	1.18	1	0.817
sMCLA	1.454	1.222	1

data. This is why we see the ITK algorithm outperform sHBGF and sMCLA on the Yeast dataset by a large margin.

1.5.3 Performance Variation with Increasing Attributes

In this section we examine how the performances of different consensus functions change as the number of attributes used for the constituent clusterings is changed. The number of attributes is an ad-hoc measure of the quality of clustering obtained and hence the difficulty of the ensemble. In general, the lesser the number of attributes in the constituent clusterings the more the confusion in the clustering solutions obtained, and hence, the more the difficulty of obtaining a consensus labeling using these clustering solutions.

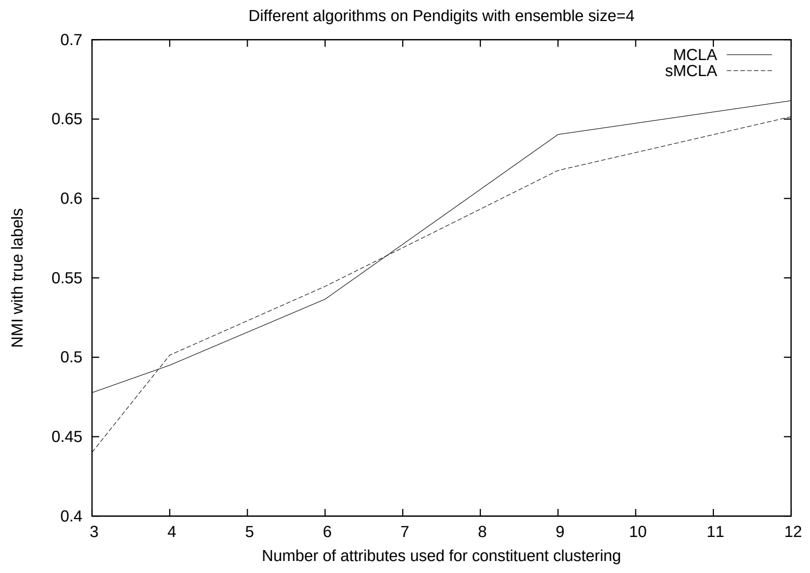
Figure 1.1 shows the variation in the performance of the existing ensemble methods and their soft variations on two datasets. The mixture of multinomial model method is not shown since its performance was much lower than the others. The datasets selected for these plots are of intermediate difficulty. As we can see, as we increase the number of attributes in the constituent clusterings the accuracy of all algorithms increases in general. For Pendigits Figure 1.1(a) only has curves for MCLA and sMCLA since we did not run HBGF and CSPA on it.

Figure 1.2 displays curves for the ITK, sHBGF, and sMCLA. As we can see the ITK algorithm outperforms the other algorithms over the whole range of attributes. But as the number of attributes is increased the accuracies of all algorithms tend to saturate.

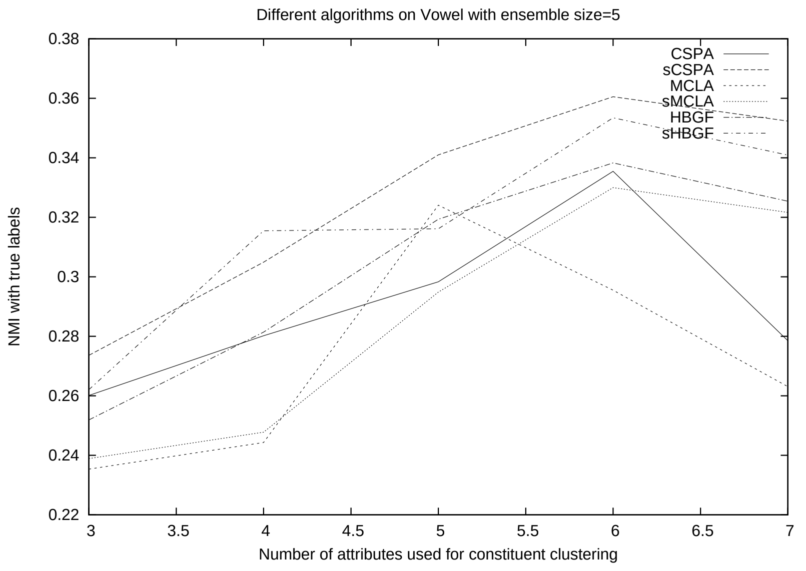
Fern and Brodley (2003) show experimentally that for high dimensional domains combining clusterings on subspace projections of the data outperforms clustering on the whole data. They also found that the impact of subspace clustering is more prominent if the number of dimensions is higher (> 60). We have not experimented with datasets that have very high dimensionality, and hence we did not observe the reduction in accuracy when using the full set of attributes.

1.5.4 Performance Variation with Increasing Ensemble Size

In this section examine the effect of increasing the number clusterings used in the ensemble on the accuracy of final clustering. Say, we set the number of attributes used to create constituent clusterings to some constant value. We would then expect that as more clusterings are added to the ensemble the combining function would have more information available to create the final clustering. This has been previously seen in the classifier ensemble literature

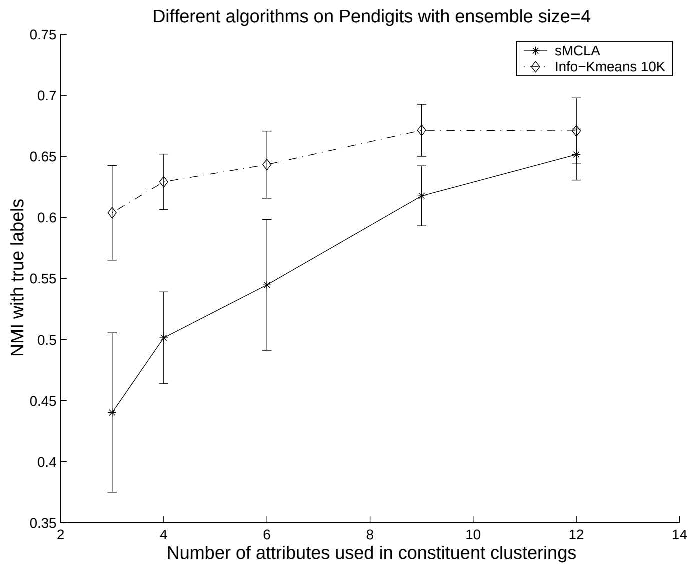


(a) Pendigits

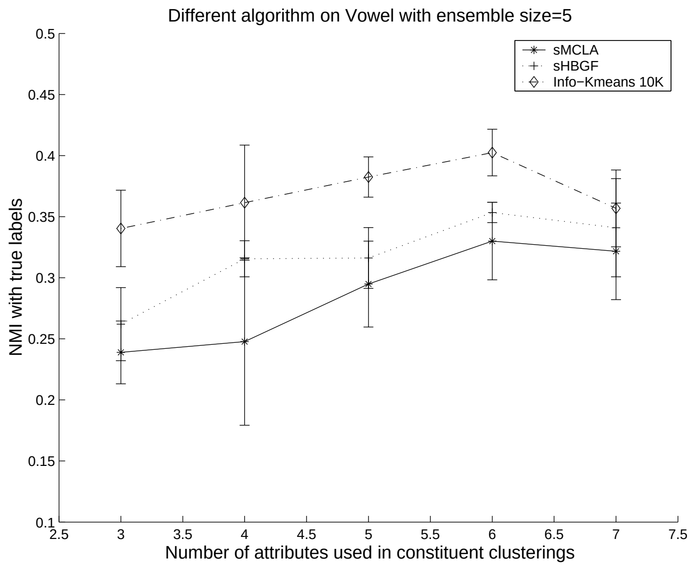


(b) Vowel

Figure 1.1 Performance of CSPA, MCLA, HBGF, sCSPA, sMCLA, and sHBGF while varying the number of attributes used in constituent clusterings



(a) Pendigits



(b) Vowel

Figure 1.2 Performance of ITK, sMCLA, and sHBGF while varying the number of attributes used in constituent clusterings

where increasing the size of the ensemble increases the accuracy until a saturation point is reached (Hansen and Salamon 1990; Melville and Mooney 2003; Opitz and Maclin 1999). Hence, the number of clusterings in an ensemble can also be said to be a measure of the difficulty of the task of combining them.

Figure 1.3 shows the variation in accuracy as number of clusterings is increased in the ensembles. We can see that as the ensembles become easier to solve the accuracy of all algorithms increases. We can also see that the increasing accuracy of most algorithms reaches a plateau once the number of clusterings grows very large. Figure 1.4 shows the variation in accuracy of the ITK, sMCLA, and sHBGF over the Pendigits and Vowel dataset as we increase the size of the ensembles. The accuracy of all the algorithms rises but the ITK algorithm performs significantly better than the others.

1.6 Conclusions and Future Work

In this chapter we presented several approaches to solving ensembles of soft clusterings. We introduced a new approach based on Information-Theoretic KMeans, and also presented simple extensions of existing approaches for hard ensembles (like sCSPA, sMCLA, and sHBGF), These approaches were extensively evaluated using datasets and ensembles of varying degrees of difficulty. Some principal conclusions we made were that soft ensembles contain useful information that can be exploited by our algorithms to obtain better consensus clusterings, especially in situations where the constituent clusterings are not very accurate. Also, ITK significantly outperforms existing approaches over most datasets, with the improvement in performance is especially large when dealing with *tough* ensembles.

Though the experimental results given in this chapter all assume the same number of clusters in each solution, the approaches do allow for varying resolution in the individual solutions. Moreover, the match of the consensus solution at different resolutions with respect to the individual solutions along the lines of (Ghosh et al. 2002) provides a good way of model selection. A challenge to the readers of this book is to identify scenarios where the use of soft ensembles provides significantly improved performance over hard ensembles, and if needed devise specialized algorithms to deal with these domains.

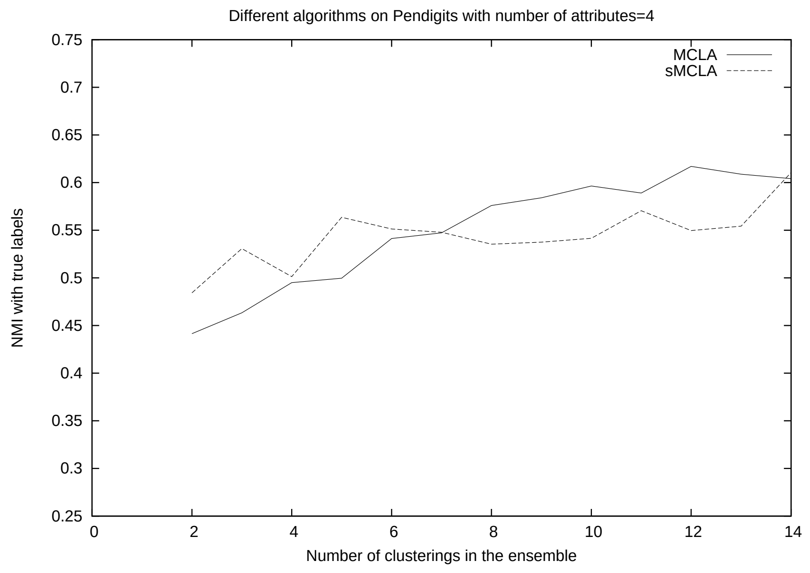
While partitioning instances we can also imagine a grouping of the clusters into meta-clusters. Algorithms based on MCLA and HBGF already compute this co-clusterings, albeit using graph partitioning based approaches. There is a significant body of research on Co-clustering or Bi-clustering using other approaches (Dhillon et al. 2003a; Madeira and Oliveira 2004), and it will be worthwhile to investigate specialized co-clustering approaches for obtaining a consensus of soft clusterings.

Bibliography

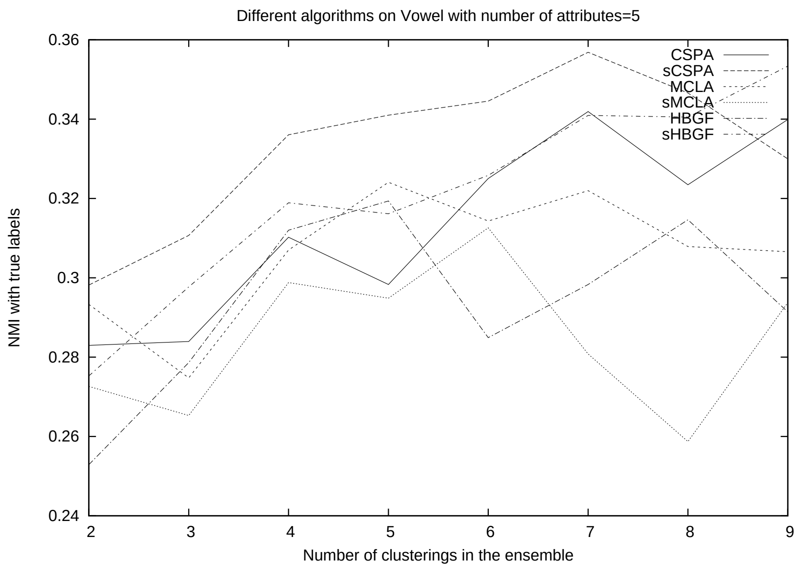
Bezdek JC and Pal S 1992 *Fuzzy Models for Pattern Recognition*. IEEE Press, Piscataway, NJ.

Blake C and Merz C 1998 UCI repository of machine learning databases, http://www.ics.uci.edu/∼mlearn/mlrepository.html.

Breiman L 1999 Combining predictors In *Combining Artificial Neural Nets* (ed. Sharkey A) Springer-Verlag pp. 31–50.

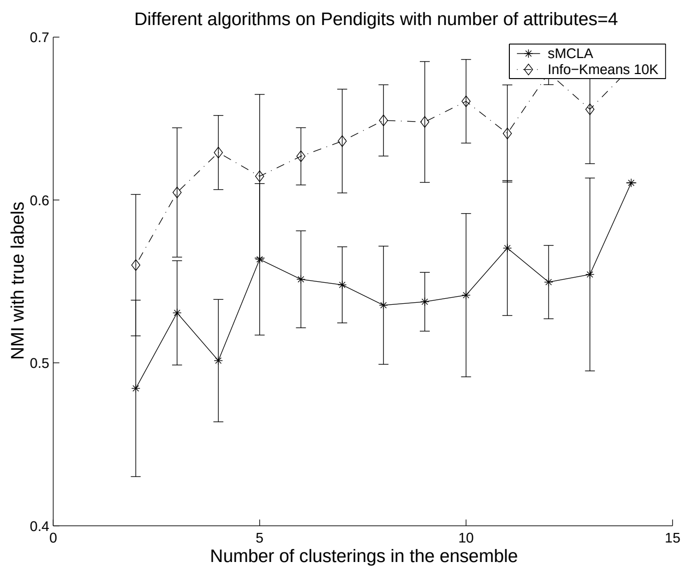


(a) Pendigits

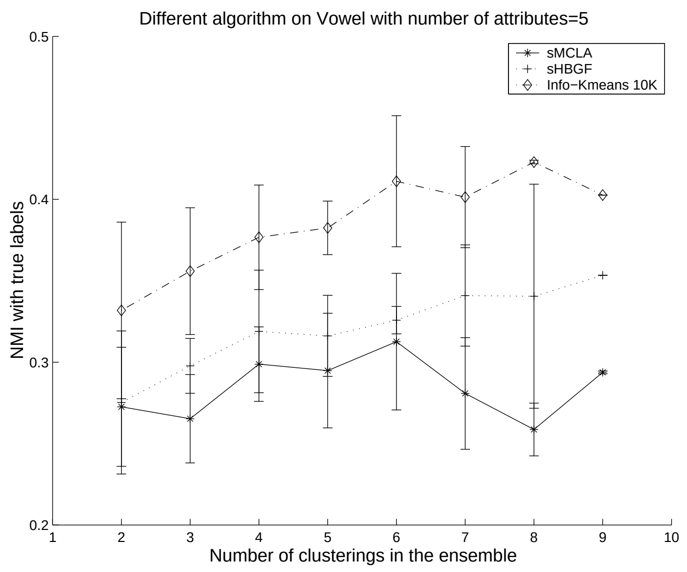


(b) Vowel

Figure 1.3 Performance of CSPA, MCLA, HBGF, sCSPA, sMCLA, and sHBGF while varying the number of constituent clusterings



(a) Pendigits



(b) Vowel

Figure 1.4 Performance of ITK, sMCLA, and sHBGF while varying the number of constituent clusterings

Dempster A, Laird N and Rubin D 1977 Maximum likelihood from incomplete data via the em algorithm *Journal of the Royal Statistical Society*, vol. 39 Series B, pp. 1–38.

Dhillon I, Mallela S and Modha D 2003a Information-Theoretic co-clustering *Proceedings of The Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD)*, pp. 89–98.

Dhillon IS 2001 Co-clustering documents and words using bipartite spectral graph partitioning *Proceedings of The Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD)*, pp. 269–274.

Dhillon IS, Mallela S and Kumar R 2003b A divisive Information-Theoretic feature clustering algorithm for text classification. *Journal of Machine Learning Research* **3**, 1265–1287.

Dimitriadou E, Weingessel A and Hornik K 2001 Voting-Merging: An ensemble method for clustering In *Proceedings of the International Conference on Artificial Neural Networks (ICANN 01)* (ed. Dorffner G, Bischof H and Hornik K), vol. LNCS 2130, pp. 217–224, Vienna, Austria.

Dunn JC 1973 A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. *Journal of Cybernetics* **3**, 32–57.

Fern XZ and Brodley CE 2003 Random projection for high dimensional clustering: A cluster ensemble approach *Proceedings of the Twentieth International Conference on Machine Learning*. ACM Press.

Fern XZ and Brodley CE 2004 Solving cluster ensemble problems by bipartite graph partitioning *Proceedings of the Twenty-first International Conference on Machine Learning*. ACM Press.

Fred A and Jain AK 2001 Finding consistent clusters in data partitions In *Proceedings of the Third International Workshop on Multiple Classifier Systems* (ed. F. Roli JK), vol. LNCS 2364, pp. 309–318.

Fred A and Jain AK 2002 Data clustering using evidence accumulation *Proceedings of the Sixteenth International Conference on Pattern Recognition (ICPR)*, pp. 276–280.

Freund Y and Schapire R 1996 Experiments with a new boosting algorithm *Proceedings of the Thirteenth International Conference on Machine Learning*, pp. 148–156. Morgan Kaufmann.

Frossyniotis DS, Pertselakis M and Stafylopatis A 2002 A multi-clustering fusion algorithm *Proceedings of the Second Hellenic Conference on AI*, pp. 225–236. Springer-Verlag.

Gablentz W, Koppen M and Dimitriadou E 2000 Robust clustering by evolutionary computation *Proc. 5th Online World Conference on Soft Computing in Industrial Applications*.

Ghosh J 2002 Multiclassifier systems: Back to the future (invited paper) In *Multiple Classifier Systems* (ed. Roli F and Kittler J) LNCS Vol. 2364, Springer pp. 1–15.

Ghosh J, Strehl A and Merugu S 2002 A consensus framework for integrating distributed clusterings under limited knowledge sharing *Proceedings of NSF Workshop on Next Generation Data Mining*, pp. 99–108.

Hadjitodorov S, Kuncheva L and Todorova L 2006 Moderate diversity for better cluster ensembles. *Information Fusion* **7(3)**, 264–275.

Hansen L and Salamon P 1990 Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **12**, 993–1001.

Hubert L and Arabie P 1985 Comparing partitions. *Journal of Classification* **2**, 193–218.

Karypis G, Aggarwal R, Kumar V and Shekhar S 1997 Multilevel hypergraph partitioning: application in VLSI domain *Proceedings of the Thirty-fourth Annual Conference on Design Automation*, pp. 526–529.

Karypis G and Kumar V 1998 A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing* **20(1)**, 359–392.

Kreiger AM and Green P 1999 A generalized rand-index method for consensus clustering of separate partitions of the same data base. *Journal of Classification* **16**, 63–89.

Kullback S and Leibler RA 1951 On information and sufficiency. *Annals of Mathematical Statistics* **22**, 79–86.

Kumar S, Ghosh J and Crawford MM 2001 Best basis feature extraction algorithms for classification of hyperspectral data *IEEE Transactions on Geoscience and Remote Sensing, Spl. Issue on Analysis of Hyperspectral Data*, vol. 39(7), pp. 1368–1379.

Kuncheva L and Hadjitodorov S 2004 Using diversity in cluster ensembles *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, pp. 1214–1219.

Madeira SC and Oliveira AL 2004 Biclustering algorithms for biological data analysis: a survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* **1(1)**, 24–45.

Melville P and Mooney RJ 2003 Constructing diverse classifier ensembles using artificial training examples *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 505–510.

Merugu S and Ghosh J 2003 Privacy-preserving distributed clustering using generative models *Proceedings of The Third IEEE International Conference on Data Mining (ICDM)*, pp. 211–218.

Ng A, Jordan M and Weiss Y 2001 On spectral clustering: Analysis and an algorithm *Proceedings of Advances in Neural Information Processing Systems 14*, pp. 849–856.

Opitz D and Maclin R 1999 Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research* **11**, 169–198.

Pedrycz W 2002 Collaborative fuzzy clustering. *Pattern Recognition Letters* **23**(14), 1675–86.

Strehl A and Ghosh J 2002 Cluster ensembles – a knowledge reuse framework for combining multiple partitions. *Journal on Machine Learning Research (JMLR)* **3**, 583–617.

Topchy A, Jain A and Punch W 2004 Mixture Model for Clustering Ensembles *Proceedings of The Fourth SIAM Conference on Data Mining (SDM)*, pp. 379–390.

Webb GI 2000 Multiboosting: A technique for combining boosting and wagging. *Machine Learning* **40**(2), 159–196.