Journal of Machine Learning Research 3 (2002) 583-617 Submitted 4/02; Published 12/02

Cluster Ensembles – A Knowledge Reuse Framework for Combining Multiple Partitions

Alexander Strehl ALEXANDER@STREHL.COM

Joydeep Ghosh GHOSH@ECE.UTEXAS.EDU
Department of Electrical and Computer Engineering
The University of Texas at Austin
Austin, TX 78712, USA

Editor: Claire Cardie

Abstract

This paper introduces the problem of combining multiple partitionings of a set of objects into a single consolidated clustering *without* accessing the features or algorithms that determined these partitionings. We first identify several application scenarios for the resultant 'knowledge reuse' framework that we call *cluster ensembles*. The cluster ensemble problem is then formalized as a combinatorial optimization problem in terms of shared mutual information. In addition to a direct maximization approach, we propose three effective and efficient techniques for obtaining high-quality combiners (consensus functions). The first combiner induces a similarity measure from the partitionings and then reclusters the objects. The second combiner is based on hypergraph partitioning. The third one collapses groups of clusters into meta-clusters which then compete for each object to determine the combined clustering. Due to the low computational costs of our techniques, it is quite feasible to use a supra-consensus function that evaluates all three approaches against the objective function and picks the best solution for a given situation. We evaluate the effectiveness of cluster ensembles in three qualitatively different application scenarios: (i) where the original clusters were formed based on non-identical sets of features, (ii) where the original clustering algorithms worked on non-identical sets of objects, and (iii) where a common data-set is used and the main purpose of combining multiple clusterings is to improve the quality and robustness of the solution. Promising results are obtained in all three situations for synthetic as well as real data-sets.

Keywords: cluster analysis, clustering, partitioning, unsupervised learning, multi-learner systems, ensemble, mutual information, consensus functions, knowledge reuse

1. Introduction

The notion of integrating multiple data sources and/or learned models is found in several disciplines, for example, the combining of estimators in econometrics (Granger, 1989), evidences in rule-based systems (Barnett, 1981) and multi-sensor data fusion (Dasarathy, 1994). A simple but effective type of multi-learner system is an *ensemble* in which each component learner (typically a regressor or classifier) tries to solve the same task. While early studies on combining multiple rankings, such as the works by Borda and Condorcet, pre-date the French Revolution (Ghosh, 2002a), this area noticeably came to life in the past decade, and now even boasts its own series of dedicated workshops (Kittler and Roli, 2002). Until now the main goal of ensembles has been to improve the accuracy and robustness of a given classification or regression task, and spectacular improvements have been obtained for a wide variety of data sets (Sharkey, 1999).

Unlike classification or regression settings, there have been very few approaches proposed for combining multiple clusterings.[1] Notable exceptions include:

- strict consensus clustering for designing evolutionary trees, typically leading to a solution at a much lower resolution than that of the individual solutions, and

- combining the results of several clusterings of a given data-set, where each solution resides in a common, known feature space, for example, combining multiple sets of cluster centers obtained by using k-means with different initializations (Bradley and Fayyad, 1998).

In this paper, we introduce the problem of *combining* multiple *partitionings* of a set of objects *without accessing the original features*. We call this the *cluster ensemble* problem, and will motivate this new, constrained formulation shortly. Note that since the combiner can only examine the cluster label but not the original features, this is a framework for knowledge reuse (Bollacker and Ghosh, 1999). The cluster ensemble design problem is more difficult than designing *classifier ensembles* since cluster labels are symbolic and so one must also solve a correspondence problem. In addition, the number and shape of clusters provided by the individual solutions may vary based on the clustering method as well as on the particular view of the data available to that method. Moreover, the desired number of clusters is often not known in advance. In fact, the 'right' number of clusters in a data-set often depends on the *scale* at which the data is inspected, and sometimes equally valid (but substantially different) answers can be obtained for the same data (Chakaravathy and Ghosh, 1996).

We call a particular clustering algorithm with a specific view of the data a *clusterer*. Each clusterer outputs a *clustering* or *labeling*, comprising the group labels for some or all objects. Some clusterers may provide additional information such as descriptions of cluster means, but we shall not use such information in this paper. There are two primary motivations for developing cluster ensembles as defined above: to exploit and reuse existing knowledge implicit in legacy clusterings, and to enable clustering over distributed data-sets in cases where the raw data cannot be shared or pooled together because of restrictions due to ownership, privacy, storage, etc. Let us consider these two application domains in greater detail.

Knowledge Reuse. In several applications, a variety of clusterings for the objects under consideration *may already exist*, and one desires to either integrate these clusterings into a single solution, or use this information to influence a new clustering (perhaps based on a different set of features) of these objects. Our first encounter with this application scenario was when clustering visitors to an e-tailing website based on market basket analysis, in order to facilitate a direct marketing campaign (Strehl and Ghosh, 2000). The company already had a variety of legacy customer segmentations based on demographics, credit rating, geographical region and purchasing patterns in

1. See Section 5 on related work for details.

their retail stores, etc. They were obviously reluctant to throw out all this domain knowledge, and instead wanted to reuse such pre-existing knowledge to create a single consolidated clustering. Note that since the legacy clusterings were largely provided by human experts or by other companies using proprietary methods, the information in the legacy segmentations had to be used *without* going back to the *original features* or the 'algorithms' that were used to obtain these clusterings. This experience was instrumental in our formulation of the cluster ensemble problem. Another notable aspect of this engagement was that the two sets of customers, purchasing from retail outlets and from the website respectively, had significant overlap but were not identical. Thus the cluster ensemble problem has to allow for missing labels in individual clusterings.

There are several other applications where legacy clusterings are available and a constrained use of such information is useful. For example, one may wish to combine or reconcile a clustering or categorization of web pages based on text analysis with those already available from Yahoo! or DMOZ (according to manually-crafted taxonomies), from Internet service providers according to request patterns and frequencies, and those indicated by a user's personal bookmarks according to his or her preferences. As a second example, clustering of mortgage loan applications based on the information in the application forms can be supplemented by segmentations of the applicants indicated by external sources such as the FICO scores provided by Fair Isaac.

Distributed Computing. The desire to perform distributed data mining is being increasingly felt in both government and industry. Often, related information is acquired and stored in geographically distributed locations due to organizational or operational constraints (Kargupta and Chan, 2000), and one needs to process data *in situ* as far as possible. In contrast, machine learning algorithms invariably assume that data is available in a single centralized location. One can argue that by transferring all the data to a single location and performing a series of merges and joins, one can get a single (albeit very large) flat file, and our favorite algorithms can then be used after randomizing and subsampling this file. But in practice, such an approach may not be feasible because of the computational, bandwidth and storage costs. In certain cases, it may not even be possible due to variety of real-life constraints including security, privacy, the proprietary nature of data and the accompanying ownership issues, the need for fault tolerant distribution of data and services, real-time processing requirements or statutory constraints imposed by law (Prodromidis et al., 2000). Interestingly, the severity of such constraints has become very evident of late as several government agencies attempt to integrate their databases and analytical techniques.

A cluster ensemble can be employed in 'privacy-preserving' scenarios where it is not possible to centrally collect all records for cluster analysis, but the distributed computing entities can share smaller amounts of higher level information such as cluster labels. The ensemble can be used for *feature-distributed clustering* in situations where each processor/clusterer has access to only a limited number of features or attributes of each object, i.e., it observes a particular *aspect* or *view* of the data. Aspects can be completely disjoint features or have partial overlaps. In gene function prediction, separate gene clusterings can be obtained from diverse sources such as gene sequence comparisons, combinations of DNA microarray data from many independent experiments, and mining of the biological literature such as MEDLINE.

An orthogonal scenario is *object-distributed clustering*, wherein each processor/clusterer has access to only a subset of all objects, and can thus only cluster the observed objects. For example, corporations tend to split their customers regionally for more efficient management. Analysis such as clustering is often performed locally, and a cluster ensemble provides a way of obtaining a holistic analysis without complete integration of the local data warehouses.

One can also consider the use of cluster ensembles for the same reasons as classification ensembles, namely to *improve the quality and robustness* of results. For classification or regression problems, it has been analytically shown that the gains from using ensemble methods involving strong learners are directly related to the amount of diversity among the individual component models (Krogh and Vedelsby, 1995, Tumer and Ghosh, 1999). One desires that each individual model be powerful, but at the same time, these models should have different inductive biases and thus generalize in distinct ways (Dietterich, 2001). So it is not surprising that ensembles are most popular for integrating relatively unstable models such as decision trees and multi-layered perceptrons. If diversity is indeed found to be beneficial in the clustering context, then it can be created in numerous ways, including:

- using different features to represent the objects. For example, images can be represented by their pixels, histograms, location and parameters of perceptual primitives or 3D scene coordinates
- varying the number and/or location of initial cluster centers in iterative algorithms such as k-means
- varying the order of data presentation in on-line methods such as BIRCH
- using a portfolio of very different clustering algorithms such as density based, k-means or soft variants such as fuzzy c-means, graph partitioning based, statistical mechanics based, etc.

It is well known that the comparative performance of different clustering methods can vary significantly across data-sets. For example, the popular k-means algorithm performs miserably in several situations where the data cannot be accurately characterized by a mixture of k Gaussians with identical covariance matrices (Karypis et al., 1999). In fact, for difficult data-sets, comparative studies across multiple clustering algorithms typically show much more variability in results than studies comparing the results of strong learners for classification (Richard and Lippmann, 1991). Thus there could be a potential for greater gains when using an ensemble for the purpose of improving clustering quality. Note that, in contrast to the knowledge reuse and distributed clustering scenarios, in this situation the combination mechanism could have had access to the original features. Our restriction that the consensus mechanism can only use cluster labels is in this case solely to simplify the problem and limit the scope of the solution, just as combiners of multiple classifiers are often based solely on the classifier outputs (for example, voting and averaging methods), although a richer design space is available.

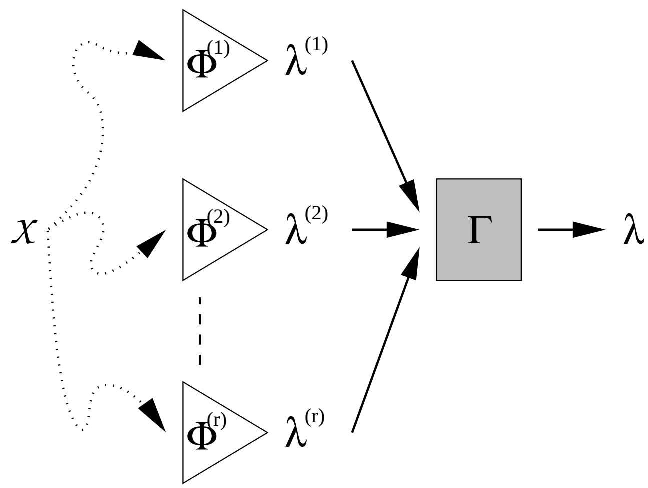


Figure 1: The Cluster Ensemble. A consensus function Γ combines clusterings $\lambda^{(q)}$ from a variety of sources, without resorting to the original object features $\mathcal{X}$ or algorithms Φ.

A final, related motivation for using a cluster ensemble is to build a *robust* clustering portfolio that can perform well over a wide range of data-sets with little hand-tuning. For example, by using an ensemble that includes approaches such as k-means, SOM (Kohonen, 1995) and DBSCAN (Ester et al., 1996), that typically work well in low-dimensional metric spaces, as well as algorithms tailored for high-dimensional sparse spaces such as spherical k-means (Dhillon and Modha, 2001) and Jaccard-based graph-partitioning (Strehl and Ghosh, 2000), one may perform very well in three dimensional as well as in 30000 dimensional spaces without having to switch models. This characteristic is very attractive to the general practitioner.

Notation. Let $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ denote a set of objects/samples/points. A partitioning of these n objects into k clusters can be represented as a set of k sets of objects $\{\mathcal{C}_\ell | \ell = 1, \ldots, k\}$ or as a label vector $\lambda \in \mathbb{N}^n$. A clusterer Φ is a function that delivers a label vector given a tuple of objects. Figure 1 shows the basic setup of the cluster ensemble: A set of r labelings $\lambda^{(1,\ldots,r)}$ is combined into a single labeling λ (the *consensus labeling*) using a consensus function Γ. Vector/matrix transposition is indicated with a superscript $\dagger$. A superscript in brackets denotes an index and not an exponent.

Organization. In the next section, we formally define the design of a cluster ensemble as an optimization problem and propose an appropriate objective function. In Section 3, we propose and compare three effective and efficient combining schemes, Γ, to tackle the combinatorial complexity of the problem. In Section 4 we describe applications of cluster ensembles for the scenarios described above, and show results on both real and artificial data.

2. The Cluster Ensemble Problem

In this section we illustrate the problem of combining multiple clusterings, propose a suitable objective function for determining a single consensus clustering, and explore the feasibility of directly optimizing this objective function using greedy approaches.

2.1 Illustrative Example

First, we will illustrate the combining of clusterings using a simple example. Let the following label vectors specify four clusterings of the same set of seven objects (see also Table 1):

$$\lambda^{(1)} = (1,1,1,2,2,3,3)^\dagger \qquad \lambda^{(2)} = (2,2,2,3,3,1,1)^\dagger$$
$$\lambda^{(3)} = (1,1,2,2,3,3,3)^\dagger \qquad \lambda^{(4)} = (1,2,?,1,2,?,?)^\dagger$$

Inspection of the label vectors reveals that clusterings $\lambda^{(1)}$ and $\lambda^{(2)}$ are logically identical. Clustering $\lambda^{(3)}$ introduces some dispute about objects x_3 and x_5. Clustering $\lambda^{(4)}$ is quite inconsistent with all the other ones, has two groupings instead of three, and also contains missing data. Now let us look for a good combined clustering with three clusters. Intuitively, a good combined clustering should *share as much information as possible* with the given four labelings. Inspection suggests that a reasonable integrated clustering is $(2,2,2,3,3,1,1)^\dagger$ (or one of the six equivalent clusterings such as $(1,1,1,2,2,3,3)^\dagger$). In fact, after performing an exhaustive search over all 301 unique clusterings of seven elements into three groups, it can be shown that this clustering shares the maximum information with the given four label vectors (in terms that are more formally introduced in the next subsection).

This simple example illustrates some of the challenges. We have already seen that the label vector is not unique. In fact, for each unique clustering there are $k!$ equivalent representations as integer label vectors. However, only one representation satisfies the following two constraints: (i) $\lambda_1 = 1$; (ii) for all $i = 1, \ldots, n-1 : \lambda_{i+1} \leq \max_{j=1,\ldots,i}(\lambda_j) + 1$. The first constraint enforces that the first object's label is cluster 1. The second constraint assures that the cluster label λ_{i+1} of any successive object x_{i+1} either has a label that occurred before or a label that is one greater than the highest label so far. By allowing only representations that fulfill both constraints, the integer vector representation can be forced to be *unique*. Transforming the labels into this 'canonical form' solves the combining problem if all clusterings are actually the same. However, if there is any discrepancy among labelings, one has to also deal with a complex correspondence problem. In general, the number of clusters found as well as each cluster's interpretation may vary tremendously among models.

2.2 Objective Function for Cluster Ensembles

Given r groupings, with the q-th grouping $\lambda^{(q)}$ having $k^{(q)}$ clusters, a consensus function Γ is defined as a function $\mathbb{N}^{n \times r} \rightarrow \mathbb{N}^n$ mapping a set of clusterings to an integrated clustering:

$$\Gamma : \{\lambda^{(q)} \mid q \in \{1, \ldots, r\}\} \rightarrow \lambda. \tag{1}$$

Let the set of groupings $\{\lambda^{(q)} \mid q \in \{1, \ldots, r\}\}$ be denoted by $\mathbf{\Lambda}$. If there is no *a priori* information about the relative importance of the individual groupings, then a reasonable goal for the consensus answer is to seek a clustering that shares the most information with the original clusterings.

Mutual information, which is a symmetric measure to quantify the statistical information shared between two distributions (Cover and Thomas, 1991), provides a sound indication of the shared information between a pair of clusterings. Let X and Y be the random variables described by the cluster labeling $\lambda^{(a)}$ and $\lambda^{(b)}$, with $k^{(a)}$ and $k^{(b)}$ groups respectively. Let $I(X,Y)$ denote the mutual information between X and Y, and $H(X)$ denote the entropy of X. One can show that $I(X,Y)$ is a metric. There is no upper bound for $I(X,Y)$, so for easier interpretation and comparisons, a normalized version of $I(X,Y)$ that ranges from 0 to 1 is desirable. Several normalizations are possible based on the observation that $I(X,Y) \leq \min(H(X), H(Y))$. These include normalizing using the arithmetic or geometric mean of $H(X)$ and $H(Y)$. Since $H(X) = I(X,X)$, we prefer the geometric mean because of the analogy with a normalized inner product in Hilbert space. Thus the normalized mutual information (NMI)[2] used is:

$$NMI(X,Y) = \frac{I(X,Y)}{\sqrt{H(X)H(Y)}}. \tag{2}$$

One can see that $NMI(X,X) = 1$, as desired. Equation 2 needs to be estimated by the sampled quantities provided by the clusterings. Let $n_h^{(a)}$ be the number of objects in cluster $\mathcal{C}_h$ according to $\lambda^{(a)}$, and $n_\ell^{(b)}$ the number of objects in cluster $\mathcal{C}_\ell$ according to $\lambda^{(b)}$. Let $n_{h,\ell}$ denote the number of objects that are in cluster h according to $\lambda^{(a)}$ as well as in group ℓ according to $\lambda^{(b)}$. Then, from Equation 2, the normalized mutual information estimate $\phi^{(\mathrm{NMI})}$ is:

$$\phi^{(\mathrm{NMI})}(\lambda^{(a)}, \lambda^{(b)}) = \frac{\sum_{h=1}^{k^{(a)}} \sum_{\ell=1}^{k^{(b)}} n_{h,\ell} \log\left(\frac{n \cdot n_{h,\ell}}{n_h^{(a)} n_\ell^{(b)}}\right)}{\sqrt{\left(\sum_{h=1}^{k^{(a)}} n_h^{(a)} \log \frac{n_h^{(a)}}{n}\right)\left(\sum_{\ell=1}^{k^{(b)}} n_\ell^{(b)} \log \frac{n_\ell^{(b)}}{n}\right)}}. \tag{3}$$

Based on this pairwise measure of mutual information, we can now define a measure between a *set* of r labelings, $\mathbf{\Lambda}$, and a *single* labeling $\hat{\lambda}$ as the average normalized mutual information (ANMI):

$$\phi^{(\mathrm{ANMI})}(\mathbf{\Lambda}, \hat{\lambda}) = \frac{1}{r} \sum_{q=1}^{r} \phi^{(\mathrm{NMI})}(\hat{\lambda}, \lambda^{(q)}). \tag{4}$$

In this paper, we propose the optimal combined clustering $\lambda^{(k-\mathrm{opt})}$ to be the one that has maximal average mutual information with all individual labelings $\lambda^{(q)}$ in $\mathbf{\Lambda}$ given that the number of consensus clusters desired is k. In other words, $\phi^{(\mathrm{ANMI})}$ is our objective function and $\lambda^{(k-\mathrm{opt})}$ is defined as

$$\lambda^{(k-\mathrm{opt})} = \arg\max_{\hat{\lambda}} \sum_{q=1}^{r} \phi^{(\mathrm{NMI})}(\hat{\lambda}, \lambda^{(q)}), \tag{5}$$

2. Our earlier work (Strehl and Ghosh, 2002a) used a slightly different normalization as only balanced clusters were desired: $NMI(X,Y) = 2 \cdot I(X,Y)/(\log k^{(a)} + \log k^{(b)})$, i.e., using arithmetic mean and assuming maximum entropy caused by perfect balancing.

where $\hat{\lambda}$ goes through all possible k-partitions (Strehl and Ghosh, 2002a). Note that this formulation treats each individual clustering equally. One can easily generalize this definition to a weighted average, which may be preferable if certain individual solutions are more important than others.

There may be situations where not all labels are known for all objects, i.e., there is missing data in the label vectors. For such cases, the consensus clustering objective from Equation 5 can be generalized by computing a weighted average of the mutual information with the known labels, with the weights proportional to the comprehensiveness of the labelings as measured by the fraction of known labels. Let $\mathcal{L}^{(q)}$ be the set of object indices with known labels for the q-th labeling. Then, the generalized objective function becomes:

$$\lambda^{(k-\mathrm{opt})} = \arg\max_{\hat{\lambda}} \sum_{q=1}^{r} |\mathcal{L}^{(q)}| \phi^{(\mathrm{NMI})}(\hat{\lambda}_{\mathcal{L}^{(q)}}, \hat{\lambda}^{(q)}_{\mathcal{L}^{(q)}}). \tag{6}$$

2.3 Direct and Greedy Optimization Approaches

The objective functions in Equations 5 and 6 represent difficult combinatorial optimization problems. An exhaustive search through all possible clusterings with k labels for the one with the maximum ANMI is formidable since for n objects and k partitions there are $\frac{1}{k!}\sum_{\ell=1}^{k} \binom{k}{\ell} (-1)^{k-\ell}\ell^{n}$ possible clusterings, or approximately $k^n/k!$ for $n \gg k$ (Jain and Dubes, 1988). For example, there are 171,798,901 ways to form four groups of 16 objects.

A variety of well known greedy search techniques, including simulated annealing and genetic algorithms, can be tried to find a reasonable solution. We have not investigated such approaches in detail since we expect the consensus ensemble problem to be mostly applied large data-sets, so computationally expensive approaches become unattractive. However, to get a feel for the quality-time tradeoffs involved, we devised and studied the following greedy optimization scheme that operates through single label changes:

The most representative single labeling (indicated by highest ANMI with all r labelings) is used as the initial labeling for the greedy algorithm. Then, for each object, the current label is changed to each of the other $k-1$ possible labels and the ANMI objective is re-evaluated. If the ANMI increases, the object's label is changed to the best new value and the algorithm proceeds to the next object. When all objects have been checked for possible improvements, a sweep is completed. If at least one label was changed in a sweep, we initiate a new sweep. The algorithm terminates when a full sweep does not change any labels, thereby indicating that a local optimum is reached. The algorithm can be readily modified to probabilistically accept decreases in ANMI as well, as in a Boltzmann machine.

As with all local optimization procedures, there is a strong dependency on the initialization. Running this greedy search starting with a random labeling is often computationally intractable, and tends to result in poor local optima. Even with an initialization that is close to an optimum, computation can be extremely slow due to exponential time complexity. Experiments with $n = 400$, $k = 10$, $r = 8$ typically averaged one hour per run on a 1 GHz PC using our implementation. Therefore results of the greedy approach are only shown in Section 3.5.

The next section proposes three algorithms that are far more efficient than the greedy approach and deliver a similar quality of results. These algorithms have been developed from intuitive heuristics rather than from the vantage point of a direct maximization.

3. Efficient Consensus Functions

In this section, we introduce three efficient heuristics to solve the cluster ensemble problem. All algorithms approach the problem by first transforming the set of clusterings into a hypergraph representation.

Cluster-based Similarity Partitioning Algorithm (CSPA). A clustering signifies a relationship between objects in the same cluster and can thus be used to establish a measure of pairwise similarity. This *induced similarity measure* is then used to recluster the objects, yielding a combined clustering.

HyperGraph Partitioning Algorithm (HGPA). In this algorithm, we approximate the maximum mutual information objective with a constrained *minimum cut* objective. Essentially, the cluster ensemble problem is posed as a partitioning problem of a suitably defined hypergraph where hyperedges represent clusters.

Meta-CLustering Algorithm (MCLA). Here, the objective of integration is viewed as a *cluster correspondence problem*. Essentially, groups of clusters (meta-clusters) have to be identified and consolidated.

The following four subsections describe the common hypergraph representation, CSPA, HGPA, and MCLA. Section 3.5 discusses differences in the algorithms and evaluates their performance in a controlled experiment.

3.1 Representing Sets of Clusterings as a Hypergraph

The first step for both of our proposed consensus functions is to transform the given cluster label vectors into a suitable hypergraph representation. In this subsection, we describe how any set of clusterings can be mapped to a hypergraph. A hypergraph consists of vertices and hyperedges. An edge in a regular graph connects exactly two vertices. A hyperedge is a generalization of an edge in that it can connect any *set* of vertices.

For each label vector $\lambda^{(q)} \in \mathbb{N}^n$, we construct the binary membership indicator matrix $\mathbf{H}^{(q)}$, with a column for each cluster (now represented as a hyperedge), as illustrated in Table 1.[3] All entries of a row in the binary membership indicator matrix $\mathbf{H}^{(q)}$ add to 1, if the row corresponds to an object with *known* label. Rows for objects with unknown label are all zero.

The concatenated block matrix $\mathbf{H} = \mathbf{H}^{(1,\ldots,r)} = (\mathbf{H}^{(1)} \ \ldots \ \mathbf{H}^{(r)})$ defines the adjacency matrix of a hypergraph with n vertices and $\sum_{q=1}^{r} k^{(q)}$ hyperedges. Each column vector $\mathbf{h}_a$ specifies a hyperedge h_a, where 1 indicates that the vertex corresponding to the row is part of that hyperedge and 0 indicates that it is not. Thus, we have mapped each cluster to a hyperedge and the set of clusterings to a hypergraph.

3. When generalizing our algorithms to *soft* clustering, $\mathbf{H}^{(q)}$ simply contains the posterior probabilities of cluster membership.

	$\lambda^{(1)}$	$\lambda^{(2)}$	$\lambda^{(3)}$	$\lambda^{(4)}$
x_1	1	2	1	1
x_2	1	2	1	2
x_3	1	2	2	?
x_4	2	3	2	1
x_5	2	3	3	2
x_6	3	1	3	?
x_7	3	1	3	?

$\Leftrightarrow$

	$\mathbf{H}^{(1)}$			$\mathbf{H}^{(2)}$			$\mathbf{H}^{(3)}$			$\mathbf{H}^{(4)}$	
	$\mathbf{h}_1$	$\mathbf{h}_2$	$\mathbf{h}_3$	$\mathbf{h}_4$	$\mathbf{h}_5$	$\mathbf{h}_6$	$\mathbf{h}_7$	$\mathbf{h}_8$	$\mathbf{h}_9$	$\mathbf{h}_{10}$	$\mathbf{h}_{11}$
v_1	1	0	0	0	1	0	1	0	0	1	0
v_2	1	0	0	0	1	0	1	0	0	0	1
v_3	1	0	0	0	1	0	0	1	0	0	0
v_4	0	1	0	0	0	1	0	1	0	1	0
v_5	0	1	0	0	0	1	0	0	1	0	1
v_6	0	0	1	1	0	0	0	0	1	0	0
v_7	0	0	1	1	0	0	0	0	1	0	0

Table 1: Illustrative cluster ensemble problem with $r = 4$, $k^{(1,\ldots,3)} = 3$, and $k^{(4)} = 2$: Original label vectors (left) and equivalent hypergraph representation with 11 hyperedges (right). Each cluster is transformed into a hyperedge.

3.2 Cluster-based Similarity Partitioning Algorithm (CSPA)

Based on a coarse resolution viewpoint that two objects have a similarity of 1 if they are in the same cluster and a similarity of 0 otherwise, a $n \times n$ binary similarity matrix can be readily created for each clustering. The entry-wise average of r such matrices representing the r sets of groupings yields an overall similarity matrix $\mathbf{S}$ with a finer resolution. The entries of $\mathbf{S}$ denote the fraction of clusterings in which two objects are in the same cluster, and can be computed in one sparse matrix multiplication $\mathbf{S} = \frac{1}{r}\mathbf{H}\mathbf{H}^{\dagger}$.[4] Figure 2 illustrates the generation of the cluster-based similarity matrix for the example given in Table 1.

Now, we can use the similarity matrix to recluster the objects using any reasonable similarity-based clustering algorithm. We chose to partition the induced similarity graph (vertex = object, edge weight = similarity) using METIS (Karypis and Kumar, 1998) because of its robust and scalable properties.

CSPA is the simplest and most obvious heuristic, but its computational and storage complexity are both quadratic in n, as opposed to the next two approaches that are near linear in n.

3.3 HyperGraph-Partitioning Algorithm (HGPA)

The second algorithm is a direct approach to cluster ensembles that re-partitions the data using the given clusters as indications of strong bonds. The cluster ensemble problem is formulated as *partitioning* the hypergraph by cutting a *minimal* number of hyperedges. We call this approach the hypergraph-partitioning algorithm (HGPA). All hyperedges are considered to have the same weight. Also, all vertices are equally weighted. Note that this includes n_ℓ-way relationship information, while CSPA only considers pairwise relationships. Now, we look for a hyperedge separator that partitions the hypergraph (Figure 3) into k unconnected components of approximately the same size. Note that obtaining comparable sized partitions is a standard constraint in graph-partitioning based clustering approaches

4. This approach can be extended to soft clusterings by using the objects' posterior probabilities of cluster membership in $\mathbf{H}$.

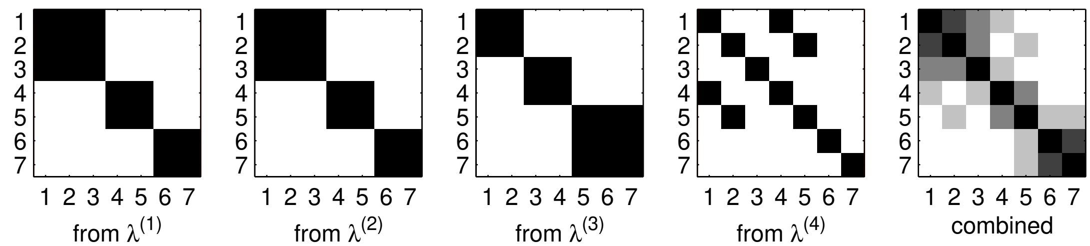


Figure 2: Illustration of cluster-based similarity partitioning algorithm (CSPA) for the cluster ensemble example problem given in Table 1. Each clustering contributes a similarity matrix. Matrix entries are shown by darkness proportional to similarity. Their average is then used to recluster the objects to yield consensus.

as it avoids trivial partitions (Karypis et al., 1999, Strehl and Ghosh, 2002b). On the other hand this means that if the natural data clusters are highly imbalanced, a graph-partitioning based approach is not appropriate. For the results of this paper, we maintain a vertex imbalance of at most 5% by imposing the following constraint: $k \cdot \max_{\ell \in \{1,\ldots,k\}} \frac{n_\ell}{n} \leq 1.05$.

Hypergraph partitioning is a well-studied area (Kernighan and Lin, 1970, Alpert and Kahng, 1995) and algorithm details are omitted here for brevity. We use the hypergraph partitioning package HMETIS (Karypis et al., 1997). HMETIS gives high-quality partitions and is very scalable. However, please note that hypergraph partitioning in general has no provision for partially cut hyperedges. This means that there is no sensitivity to how much of a hyperedge is left in the same group after the cut. This can be problematic for our applications. Let us consider the example from Table 1. For simplicity, let us assume that only the three hyperedges from $\lambda^{(1)}$ are present. The two partitionings $\{\{x_1, x_2, x_7\}, \{x_3, x_4\}, \{x_5, x_6\}\}$ and $\{\{x_1, x_7\}, \{x_3, x_4\}, \{x_2, x_5, x_6\}\}$ both cut all three hyperedges. The first is intuitively superior, because 2/3 of the hyperedge $\{x_1, x_2, x_3\}$ 'remains' versus only 1/3 in the second. However, in standard hypergraph partitioning they have equivalent quality since both cut the same number of hyperedges.

3.4 Meta-CLustering Algorithm (MCLA)

In this subsection, we introduce the third algorithm to solve the cluster ensemble problem. The Meta-CLustering Algorithm (MCLA) is based on clustering clusters. It also yields object-wise confidence estimates of cluster membership.

We represent each cluster by a hyperedge. The idea in MCLA is to group and collapse related hyperedges and assign each object to the collapsed hyperedge in which it participates most strongly. The hyperedges that are considered related for the purpose of collapsing are determined by a graph-based clustering of hyperedges. We refer to each cluster of hyperedges as a meta-cluster $\mathcal{C}^{(\mathrm{M})}$. Collapsing reduces the number of hyperedges from $\sum_{q=1}^{r} k^{(q)}$ to k. The detailed steps are:

Construct Meta-graph. Let us view all the $\sum_{q=1}^{r} k^{(q)}$ indicator vectors $\mathbf{h}$ (the hyperedges of $\mathbf{H}$) as vertices of another regular undirected graph, the meta-graph. The edge

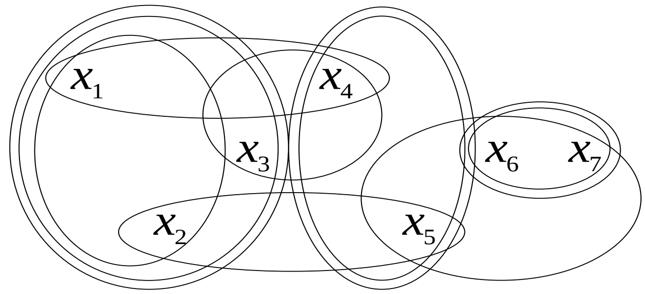


Figure 3: Illustration of hypergraph partitioning algorithm (HGPA) for the cluster ensemble example problem given in Table 1. Each hyperedge is represented by a closed curve enclosing the vertices it connects. The combined clustering $\{\{x_1, x_2, x_3\}, \{x_4, x_5\}, \{x_6, x_7\}\}$ has the minimal hyperedge cut of 4 and is as balanced as possible for three clusters of seven objects.

weights are proportional to the similarity between vertices. A suitable similarity measure here is the binary Jaccard measure, since it is the ratio of the intersection to the union of the sets of objects corresponding to the two hyperedges. Formally, the edge weight $w_{a,b}$ between two vertices h_a and h_b as defined by the binary Jaccard measure of the corresponding indicator vectors $\mathbf{h}_a$ and $\mathbf{h}_b$ is: $w_{a,b} = \frac{\mathbf{h}_a^\dagger \mathbf{h}_b}{\|\mathbf{h}_a\|_2^2 + \|\mathbf{h}_b\|_2^2 - \mathbf{h}_a^\dagger \mathbf{h}_b}$.

Since the clusters are non-overlapping (hard), there are no edges among vertices of the same clustering $\mathbf{H}^{(q)}$ and, thus, the meta-graph is r-partite, as shown in Figure 4.

Cluster Hyperedges. Find matching labels by partitioning the meta-graph into k balanced meta-clusters. We use the graph partitioning package METIS in this step. This results in a clustering of the $\mathbf{h}$ vectors. Each meta-cluster has approximately r vertices. Since each vertex in the meta-graph represents a distinct cluster label, a meta-cluster represents a group of corresponding labels.

Collapse Meta-clusters. For each of the k meta-clusters, we collapse the hyperedges into a single meta-hyperedge. Each meta-hyperedge has an association vector which contains an entry for each object describing its level of association with the corresponding meta-cluster. The level is computed by averaging all indicator vectors $\mathbf{h}$ of a particular meta-cluster.[5] An entry of 0 or 1 indicates the weakest or strongest association, respectively.

Compete for Objects. In this step, each object is assigned to its most associated meta-cluster: Specifically, an object is assigned to the meta-cluster with the highest entry in the association vector. Ties are broken randomly. The confidence of an assignment is reflected by the winner's share of association (ratio of the winner's association to the sum of all other associations). Note that not every meta-cluster can be guaranteed to win at least one object. Thus, there are *at most* k labels in the final combined clustering λ.

5. A weighted average can be used if the initial clusterings have associated confidences as in soft clustering.

Figure 4 illustrates meta-clustering for the example given in Table 1 where $r = 4$, $k = 3$, $k^{(1,\ldots,3)} = 3$, and $k^{(4)} = 2$. Figure 4 shows the original 4-partite meta-graph. The three meta-clusters are indicated by symbols $\circ$, $\times$, and $+$. Consider the first meta-cluster, $\mathcal{C}_1^{(M)} = \{h_3, h_4, h_9\}$ (the $\circ$ markers in Figure 4). Collapsing the hyperedges yields the object-weighted meta-hyperedge $h_1^{(M)} = \{v_5, v_6, v_7\}$ with association vector $(0,0,0,0,1/3,1,1)^\dagger$. Subsequently, meta-cluster $\mathcal{C}_1^{(M)}$ will win the competition for vertices/objects v_6 and v_7, and thus represent the cluster $\mathcal{C}_1 = \{x_6, x_7\}$ in the resulting integrated clustering. Our proposed meta-clustering algorithm robustly outputs $(2,2,2,3,3,1,1)^\dagger$, one of the 6 optimal clusterings which is equivalent to clusterings $\lambda^{(1)}$ and $\lambda^{(2)}$. The uncertainty about some objects is reflected in the confidences 3/4, 1, 2/3, 1, 1/2, 1, and 1 for objects 1 through 7, respectively.

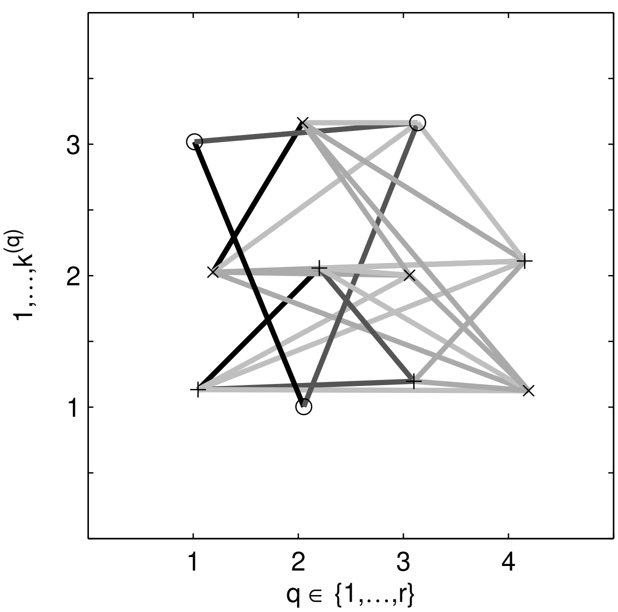


Figure 4: Illustration of Meta-CLustering Algorithm (MCLA) for the cluster ensemble example problem given in Table 1. The 4-partite meta-graph is shown. Edge darkness increases with edge weight. The vertex positions are slightly perturbed to expose otherwise occluded edges. The three meta-clusters are indicated by symbols $\circ$, $\times$, and $+$.

3.5 Discussion and Comparison

Let us first take a look at the worst-case time complexity of the proposed algorithms. Assuming quasi-linear (hyper-)graph partitioners such as (H)METIS, CSPA is $O(n^2kr)$, HGPA is $O(nkr)$, and MCLA is $O(nk^2r^2)$. The fastest is HGPA, closely followed by MCLA since k tends to be small. CSPA is slower and can be impractical for large n. The greedy approach described in the previous section is the slowest and often is intractable for large n.

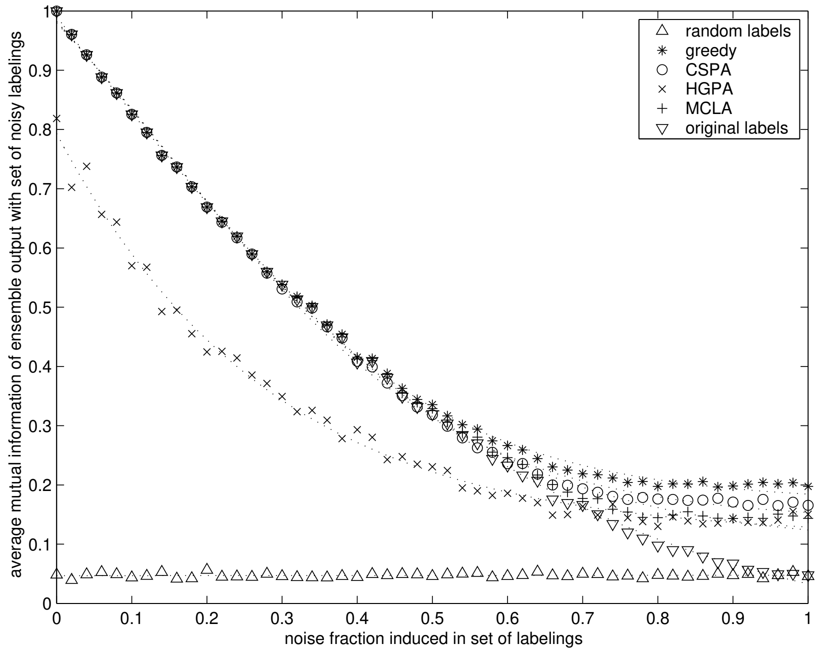
We performed a controlled experiment that allows us to compare the properties of the three proposed consensus functions. First, we partition $n = 400$ objects into $k = 10$ groups at random to obtain the original clustering κ.[6] We duplicate this clustering $r = 8$ times. Now in each of the 8 labelings, a fraction of the labels is replaced with random labels from a uniform distribution from 1 to k. Then, we feed the noisy labelings to the proposed consensus functions. The resulting combined labeling is evaluated in two ways. First, we measure the normalized objective function $\phi^{(\mathrm{ANMI})}(\mathbf{\Lambda}, \lambda)$ of the ensemble output λ with all the individual labels in $\mathbf{\Lambda}$. Second, we measure the normalized mutual information of each consensus labeling with the original undistorted labeling using $\phi^{(\mathrm{NMI})}(\kappa, \lambda)$. For better comparison, we added a random label generator as a baseline method. Also, performance measures of a hypothetical consensus function that returns the original labels are included to illustrate maximum performance for low noise settings.[7]

Figure 5 shows the results. As noise increases, labelings share less information and thus the maximum obtainable $\phi^{(\mathrm{ANMI})}(\mathbf{\Lambda}, \lambda)$ decreases, and so does $\phi^{(\mathrm{ANMI})}(\mathbf{\Lambda}, \lambda)$ for all techniques (Figure 5,top). HGPA performs the worst in this experiment, which we believe is due to its current inability to cater to partially cut edges. In low noise, both, MCLA and CSPA recover the original labelings. MCLA retains more $\phi^{(\mathrm{ANMI})}(\mathbf{\Lambda}, \lambda)$ than CSPA in the presence of medium to high noise. Interestingly, in very high noise settings CSPA exceeds MCLA's performance. Note also that for such high noise settings the original labels have a lower average normalized mutual information $\phi^{(\mathrm{ANMI})}(\mathbf{\Lambda}, \lambda)$. This is because the set of labels is almost completely random and the consensus algorithms recover whatever little common information is present, whereas the original labeling is now almost fully unrelated. Realistically, noise should not exceed 50% and MCLA seems to perform best in this simple controlled experiment. Figure 5 also illustrates how well the algorithms can recover the true labeling in the presence of noise for robust clustering. As noise increases labelings share less information with the true labeling and thus the ensemble's $\phi^{(\mathrm{NMI})}(\kappa, \lambda)$ decreases. The ranking of the algorithms is the same using this measure, with MCLA best, followed by CSPA, and HGPA worst. In fact, MCLA recovers the original labeling at up to 25% noise in this scenario! For less than 50% noise, the algorithms essentially have the same ranking regardless of whether $\phi^{(\mathrm{ANMI})}(\mathbf{\Lambda}, \lambda)$ or $\phi^{(\mathrm{NMI})}(\kappa, \lambda)$ is used. This indicates that our proposed objective function $\phi^{(\mathrm{ANMI})}(\mathbf{\Lambda}, \lambda)$ is a suitable choice in real applications where κ and hence $\phi^{(\mathrm{NMI})}(\kappa, \lambda)$ is not available.

The direct greedy optimization approach performs similar to CSPA in terms of $\phi^{(\mathrm{NMI})}(\kappa, \lambda)$ but scores lower than MCLA in most cases. In terms of $\phi^{(\mathrm{ANMI})}(\mathbf{\Lambda}, \lambda)$ the greedy approach returns a higher score than CSPA, HGPA, and MCLA only for unrealistically high (>75%) noise levels. More importantly, the greedy approach is tractable only when there are very few datapoints, dimensions, and clusters, due to its high computational complexity.

This experiment indicates that MCLA should be best suited in terms of time complexity as well as quality. In the applications and experiments described in the following sections, we observe that each combining method can result in a higher ANMI than the others

6. Labels are obtained by a random permutation. Groups are balanced.
7. In low noise settings, the original labels are the global maximum, since they share the most mutual information with the distorted labelings.



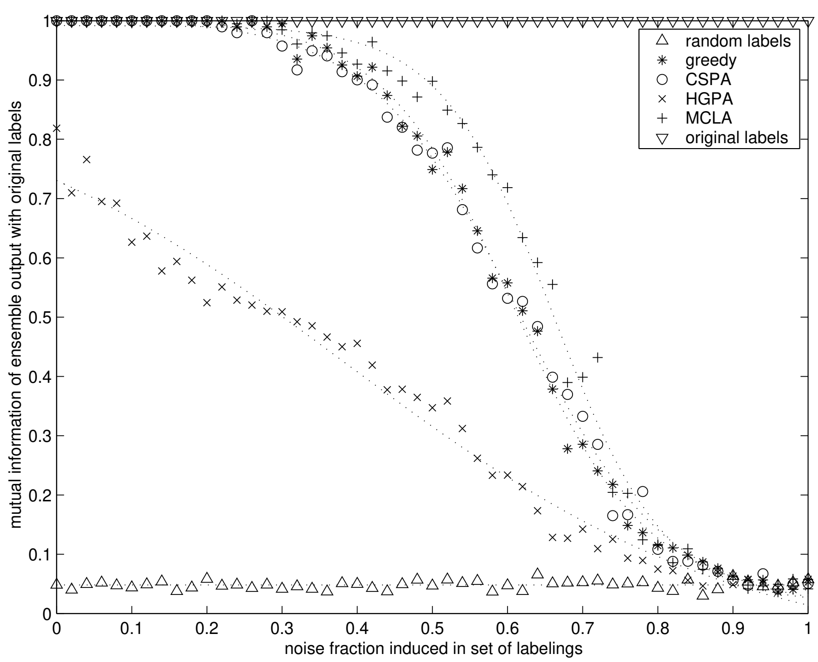


Figure 5: Comparison of consensus functions in terms of the objective function $\phi^{(\mathrm{ANMI})}(\mathbf{\Lambda}, \lambda)$ (top) and in terms of their normalized mutual information with original labels $\phi^{(\mathrm{NMI})}(\kappa, \lambda)$ (bottom) for various noise levels. A fitted sigmoid (least squared error) is shown for all algorithms to show the trend.

for particular setups. In fact, we found that MCLA tends to be best in low noise/diversity settings and HGPA/CSPA tend to be better in high noise/diversity settings. This is because MCLA assumes that there are meaningful cluster correspondences, which is more likely to be true when there is little noise and less diversity. Thus, it is useful to have all three methods.

Our objective function has the added advantage that it allows one to add a stage that selects the best consensus function without any supervisory information, by simply selecting the one with the highest ANMI. So, for the experiments in this paper, we first report the results of this 'supra'-consensus function Γ, obtained by running *all three* algorithms, CSPA, HGPA and MCLA, and selecting the one with the greatest ANMI. Then, if significant differences or notable trends are observed among the three algorithms, this further level of detail is described. Note that the supra-consensus function is completely unsupervised and avoids the problem of selecting the best combiner for a data-set beforehand.

4. Consensus Clustering Applications and Experiments

Consensus functions *enable* a variety of new approaches to several problems. After introducing the data-sets used and the evaluation methodology, we discuss in Section 4.3 how distributed clustering can be performed when each entity has access only to a very limited subset of features. Section 4.4 shows how one can operate on several very limited subsets of objects and combine them afterwards, thereby enabling distributed clustering. Finally, in Section 4.5 we illustrate how the robustness of clustering can be increased through combining a set of clusterings.

4.1 Data-Sets

We illustrate the cluster ensemble applications on two real and two artificial data-sets. Table 2 summarizes some basic properties of the data-sets (left) and parameter choices (right). The first data-set (`2D2K`) was artificially generated and contains 500 points each of two 2-dimensional (2D) Gaussian clusters with means $(-0.227, 0.077)^\dagger$ and $(0.095, 0.323)^\dagger$ and diagonal covariance matrices with 0.1 for all diagonal elements. The second data-set (`8D5K`) contains 1000 points from five multivariate Gaussian distributions (200 points each) in 8D space. Again, clusters all have the same variance (0.1), but different means. Means were drawn from a uniform distribution within the unit hypercube. Both artificial data-sets are available for download at `http://strehl.com/`.

The third data-set (`PENDIG`) is for pen-based recognition of handwritten digits. It is publicly available from the UCI Machine Learning Repository and was contributed by Alpaydin and Alimoglu. It contains 16 spatial features for each of the 7494 training and 3498 test cases (objects). There are ten classes of roughly equal size (balanced clusters) in the data corresponding to the digits 0 to 9.

The fourth data-set is for text clustering. The 20 original Yahoo! news categories in the data are `Business`, `Entertainment` (`no sub-category`, `art`, `cable`, `culture`, `film`, `industry`, `media`, `multimedia`, `music`, `online`, `people`, `review`, `stage`, `television`, `variety`), `Health`, `Politics`, `Sports` and `Technology`. The data is publicly available from `ftp://ftp.cs.umn.edu/dept/users/boley/` (K1 series) and was used in Boley et al. (1999) and Strehl et al. (2000). The raw 21839×2340 word-document matrix consists of the

name	features	#features	#categories	balance	similarity	#clusters
2D2K	real	2	2	1.00	Euclidean	2
8D5K	real	8	5	1.00	Euclidean	5
PENDIG	real	16	10	0.87	Euclidean	10
YAHOO	ordinal	2903	20	0.24	Cosine	40

Table 2: Overview of data-sets for cluster ensemble experiments. Balance is defined as the ratio of the average category size to the largest category size.

non-normalized occurrence frequencies of stemmed words, using Porter's suffix stripping algorithm (Frakes, 1992). Pruning all words that occur less than 0.01 or more than 0.10 times on average because they are insignificant (for example, `haruspex`) or too generic (for example, `new`), results in $d = 2903$. We call this data-set `YAHOO`.

For `2D2K`, `8D5K`, and `PENDIG` we use $k = 2$, 5, and 10, respectively. When clustering `YAHOO`, we use $k = 40$ clusters unless noted otherwise. We chose two times the number of categories, since this seemed to be the more natural number of clusters as indicated by preliminary runs and visualization.[8] For `2D2K`, `8D5K`, and `PENDIG` we use Euclidean-based similarity. For `YAHOO` we use cosine-based similarity.

4.2 Evaluation Criteria

Evaluating the quality of a clustering is a non-trivial and often ill-posed task. In fact, many definitions of objective functions for clusterings exist (Jain and Dubes, 1988). *Internal* criteria formulate quality as a function of the given data and/or similarities. For example, the mean squared error criterion (for k-means) and other measures of compactness are popular evaluation criteria. Measures can also be based on isolation, such as the min-cut criterion, which uses the sum of edge weights across clusters (for graph partitioning). When using internal criteria, clustering becomes an optimization problem, and a clusterer can evaluate its own performance and tune its results accordingly.

External criteria, on the other hand, impose quality by additional, external information not given to the clusterer, such as category labels. This approach is sometimes more appropriate since groupings are ultimately evaluated externally by humans. For example, when objects have already been categorized by an external source, i.e., when class labels are available, we can use information theoretic measures to quantify the match between the categorization and the clustering. Previously, average purity and entropy-based measures to assess clustering quality from 0 (worst) to 1 (best) have been used (Boley et al., 1999). While average purity is intuitive to understand, it favors small clusters, and singletons score the highest. Also, a monolithic clustering (one single cluster for all objects) receives a score as high as the fraction of objects in the biggest category. In unstratified data-sets, this number might be close to the maximum quality 1, while the intuitive quality is very low. An entropy-based measure is better than purity, but still favors smaller clusters.

Normalized mutual information provides a measure that is impartial with respect to k as compared to purity and entropy. It reaches its maximum value of 1 only when the two

8. Using a greater number of clusters than categories allows modeling of multi-modal categories.

sets of labels have a perfect one-to-one correspondence. We shall use the categorization labels κ to evaluate the cluster quality by computing $\phi^{(\mathrm{NMI})}(\kappa, \lambda)$, as defined in Equation 3.

4.3 Feature-Distributed Clustering (FDC)

In Feature-Distributed Clustering (FDC), we show how cluster ensembles can be used to boost the quality of results by combining a set of clusterings obtained from partial views of the data. As mentioned in the introduction, distributed databases that cannot be integrated at a single centralized location because of proprietary data aspects, privacy concerns, performance issues, etc., result in such scenarios. In such situations, it is more realistic to have one clusterer for each database, and transmit only the cluster labels (but not the attributes of each record) to a central location where they can be combined using the supra-consensus function.

For our experiments, we simulate such a scenario by running several clusterers, each having access to only a restricted, small subset of features (subspace). Each clusterer has a partial *view* of the data. Note that in our experiments, these views have been created from a common, full feature space of public-domain data-sets, while in a real-life scenario the different views would be determined *a priori* in an application-specific way. Each clusterer has access to all objects. The clusterers find groups in their views/subspaces using the same clustering technique. In the combining stage, individual cluster labels are integrated using our supra-consensus function.

First, let us discuss experimental results for the `8D5K` data, since they lend themselves well to illustration. We create five random 2D views (through selection of a pair of features) of the 8D data, and use Euclidean-based graph-partitioning with $k = 5$ in each view to obtain five individual clusterings. The five individual clusterings are then combined using our supra-consensus function proposed in the previous section. The clusters are linearly separable in the full 8D space. Clustering in the 8D space yields the original generative labels and is referred to as the reference clustering. In Figure 6a, the reference clustering is illustrated by coloring the data points in the space spanned by the first and second principal components (PCs). Figure 6b shows the final FDC result after combining five subspace clusterings. Each clustering has been computed from randomly selected feature pairs. These subspaces are shown in Figure 7. Each of the rows corresponds to a random selection of two out of eight feature dimensions. For each of the five chosen feature pairs, a row shows the 2D clustering (left, feature pair shown on axis) and the same 2D clustering labels used on the data projected onto the global two principal components (right). For consistent appearance of clusters across rows, the dot colors/shapes have been matched using meta-clusters. All points in clusters of the same meta-cluster share the same color/shape among all plots. In any subspace, the clusters can not be segregated well due to overlaps. The supra-consensus function can combine the partial knowledge of the 5 clusterings into a very good clustering. FDC results (Figure 6b) are clearly superior to any of the five individual results (Figure 7(right)) and are almost flawless compared to the reference clustering (Figure 6a). The best individual result has 120 'mislabeled' points while the consensus only has three points 'mislabeled'.

We also conducted FDC experiments on the other three data-sets. Table 3 summarizes the results and several comparison benchmarks. The choice of the number of random

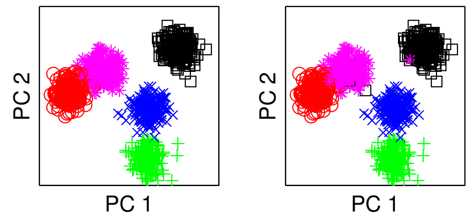


Figure 6: Reference clustering (a) and FDC consensus clustering (b) of 8D5K data. Data points are projected along the first two principal components of the 8D data. The reference clustering is obtained by graph partitioning using Euclidean similarity in original 8D space and is identical to the generating distribution assignment. The consensus clustering is derived from the combination of five clusterings, each obtained in 2D (from random feature pairs, see Figure 7). The consensus clustering (b) is clearly superior compared to any of the five individual results (Figure 7(right)) and is almost flawless compared to the reference clustering (a).

subspaces r and their dimensionality is currently driven by the user. For example, in the YAHOO case, 20 clusterings were performed in 128 dimensions (occurrence frequencies of 128 random words) each. The average quality among the results was 0.16 and the best quality was 0.20. Using the supra-consensus function to combine all 20 labelings yields a quality of 0.31, or 156% more mutual information than the average individual clustering. In all scenarios, the consensus clustering is as good as or better than the best individual input clustering, and always better than the average quality of individual clusterings. When processing on all features is not possible and only limited views exist, a cluster ensemble can boost results significantly compared to individual clusterings. However, since the combiner has no feature information, the consensus clustering tends to be poorer than the clustering on all features. As discussed in the introduction, in knowledge-reuse application scenarios, the original features are unavailable, so a comparison to an all-feature clustering is only done as a reference.

The supra-consensus function chooses either MCLA and CSPA results, but the difference is not statistically significant. As noted before MCLA is much faster and should be the method of choice if only one is needed. HGPA delivers poor ANMI for 2D2K and 8D5K but improves with more complex data in PENDIG and YAHOO. However, MCLA and CSPA performed significantly better than HGPA for all four data-sets.

4.4 Object-Distributed Clustering (ODC)

A dual of the application described in the previous section, is object-distributed clustering (ODC). In this scenario, individual clusterers have a limited selection of the object population but have access to all of the features of the objects they are provided with.

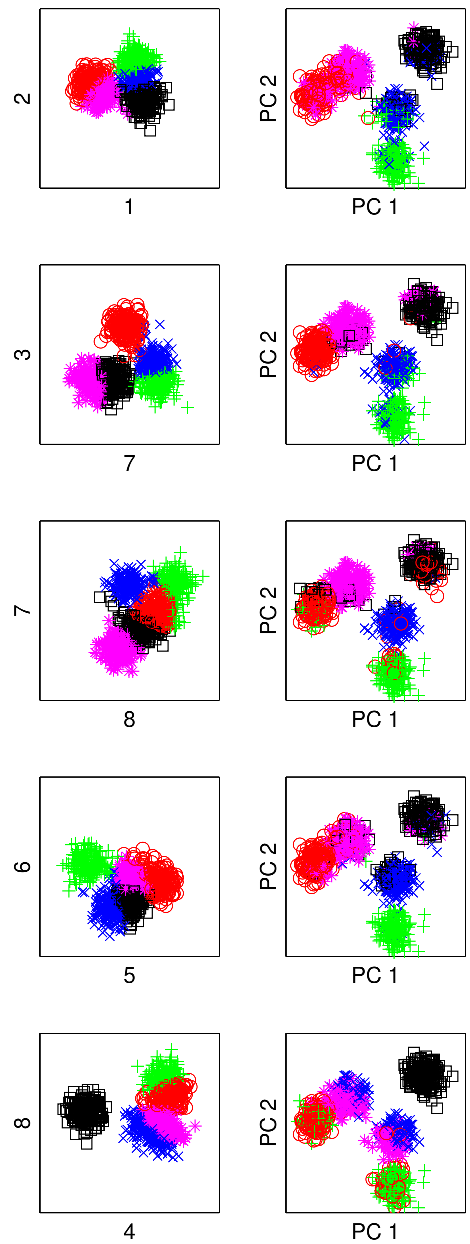


Figure 7: Illustration of feature-distributed clustering (FDC) on `8D5K` data. Each row corresponds to a random selection of two out of eight feature dimensions. For each of the five chosen feature pairs, a row shows the clustering (colored) obtained on the 2D subspace spanned by the selected feature pair (left) and visualizes these clusters on the plane of the global two principal components (right). In any subspace, the clusters can not be segregated well due to strong overlaps. The supra-consensus function can combine the partial knowledge of the five clusterings into a far superior clustering (Figure 6b).

input and parameters			quality				
data	sub-space #dims	# models r	all features $\phi^{(\mathrm{NMI})}(\kappa, \lambda^{(\mathrm{all})})$	**consensus** $\phi^{(\mathrm{NMI})}(\kappa, \lambda)$	max subspace $\max_q$ $\phi^{(\mathrm{NMI})}(\kappa, \lambda^{(q)})$	average subspace avg_q $\phi^{(\mathrm{NMI})}(\kappa, \lambda^{(q)})$	min subspace $\min_q$ $\phi^{(\mathrm{NMI})}(\kappa, \lambda^{(q)})$
`2D2K`	1	3	0.84747	**0.68864**	0.68864	0.64145	0.54706
`8D5K`	2	5	1.00000	**0.98913**	0.76615	0.69823	0.62134
`PENDIG`	4	10	0.67805	**0.63918**	0.47865	0.41951	0.32641
`YAHOO`	128	20	0.48877	**0.41008**	0.20183	0.16033	0.11143

Table 3: FDC results. The consensus clustering is as good as or better than the best individual subspace clustering.

This is somewhat more difficult than FDC, since the labelings are partial. Because there is no access to the original features, the combiner Γ needs some overlap between labelings to establish a meaningful consensus[9].

Object distribution can naturally result from operational constraints in many application scenarios. For example, datamarts of individual stores of a retail company may only have records of visitors to that store, but there are enough people who visit more than one store of that company to result in the desired overlap. On the other hand, even if all of the data is centralized, one may artificially 'distribute' them in the sense of running clustering algorithms on different but overlapping samples (record-wise partitions) of the data, and then combine the results as this can provide a computational speedup when the individual clusterers have super-linear time complexity.

In this subsection, we will discuss how one can use consensus functions on overlapping sub-samples. We propose a wrapper to any clustering algorithm that simulates a scenario with distributed objects and a combiner that does not have access to the original features. In ODC, we introduce an *object partitioning* and a corresponding *clustering merging* step. The actual clustering is referred to as the inner loop clustering. In the pre-clustering partitioning step Π, the entire set of objects $\mathcal{X}$ is decomposed into p overlapping partitions π:

$$\mathcal{X} \xrightarrow{\Pi} \{\pi^{(q)} \mid q \in \{1, \ldots, p\}\} \tag{7}$$

Two partitions $\pi^{(a)}$ and $\pi^{(b)}$ are *overlapping* if and only if $|\pi^{(a)} \cap \pi^{(b)}| > 0$. Also, a set of partitions provides *full coverage* if and only if $\mathcal{X} = \bigcup_{q=1}^{p} \pi^{(q)}$.

The ODC framework is parameterized by p, the number of partitions, and v, the repetition factor. The repetition factor $v > 1$ defines the total number of points processed in all p partitions combined to be (approximately) vn.

Let us assume that the data is not ordered, so any contiguous indexed subsample is equivalent to a random subsample. For this simulation, we decided to give every partition the same number of objects to maximize speedup. Thus, in any partition π there are $|\pi| \approx \frac{nv}{p}$ objects. Now that we have the number of objects $|\pi|$ in each partition, let us propose a simple coordinated sampling strategy: For each partition there are $\frac{n}{p}$ objects deterministically picked so that the union of all p partitions provides full coverage of all n objects. The remaining objects for a particular partition are then picked randomly without

9. If features are available, one can merge partitions based on their locations in feature space to reach consensus.

replacement from the objects not yet in that partition. There are many other ways of coordinated sampling. In this paper we will limit our discussion to this one strategy for brevity.

Each partition is processed by independent, identical clusterers (chosen appropriately for the application domain). For simplicity, we use the same number of clusters k in the sub-partitions. The post-clustering merging step is done using our supra-consensus function Γ.

$$\{\lambda^{(q)} \mid q \in \{1, \ldots, p\}\} \xrightarrow{\Gamma} \lambda \tag{8}$$

Since every partition only looks at a fraction of the data, there are missing labels in the $\lambda^{(q)}$'s. Given sufficient overlap, the supra-consensus function Γ ties the individual clusters together and delivers a consensus clustering.

We performed the following experiment to demonstrate how the ODC framework can be used to perform clustering on partially overlapping samples without access to the original features. We use graph partitioning as the clusterer in each processor. Figure 8 shows our results for the four data-sets. Each plot in Figure 8 shows the relative mutual information (fraction of mutual information retained as compared to the reference clustering on all objects and features) as a function of the number of partitions. We fix the sum of the number of objects in all partitions to be double the number of objects (repetition factor $v = 2$). Within each plot, p ranges from 2 to 72 and each ODC result is marked with a $\circ$. For each of the plots, we fitted a sigmoid function to summarize the behavior of ODC for that scenario.

Clearly, there is a tradeoff in the number of partitions versus quality. As p approaches vn, each clusterer only receives a single point and can make no reasonable grouping. For example, in the `YAHOO` case, for $v = 2$ processing on 16 partitions still retains around 80% of the full quality. For less complex data-sets, such as `2D2K`, combining 16 partial partitionings ($v = 2$) still yields 90% of the quality. In fact, `2D2K` can be clustered in 72 partitions at 80% quality. Also, we observed that for easier data-sets there is a smaller *absolute* loss in quality for more partitions p.

Regarding our proposed techniques, all three consensus algorithms achieved similar ANMI scores without significant differences for `8D5K`, `PENDIG`, and `YAHOO`. HGPA had some instabilities for the `2D2K` data-set, delivering inferior consensus clusterings compared to MCLA and HGPA.

In general, we believe the loss in quality with p has two main causes. First, through the reduction of considered pairwise relations, the problem is simplified as speedup increases. At some point too much relationship information is lost to reconstruct the original clusters. The second factor is related to the balancing constraints used by the graph partitioner in the inner loop: the sampling strategies cannot maintain the balance, so enforcing them in clustering hurts quality. A relaxed inner loop clusterer might improve results.

Distributed clustering using a cluster ensemble is particularly useful when the inner loop clustering algorithm has superlinear complexity ($> O(n)$) and a fast consensus function (such as MCLA and HGPA) is used. In this case, additional speedups can be obtained through distribution of objects. Let us assume that the inner loop clusterer has a complexity of $O(n^2)$ (for example, similarity-based approaches or efficient agglomerative clustering)

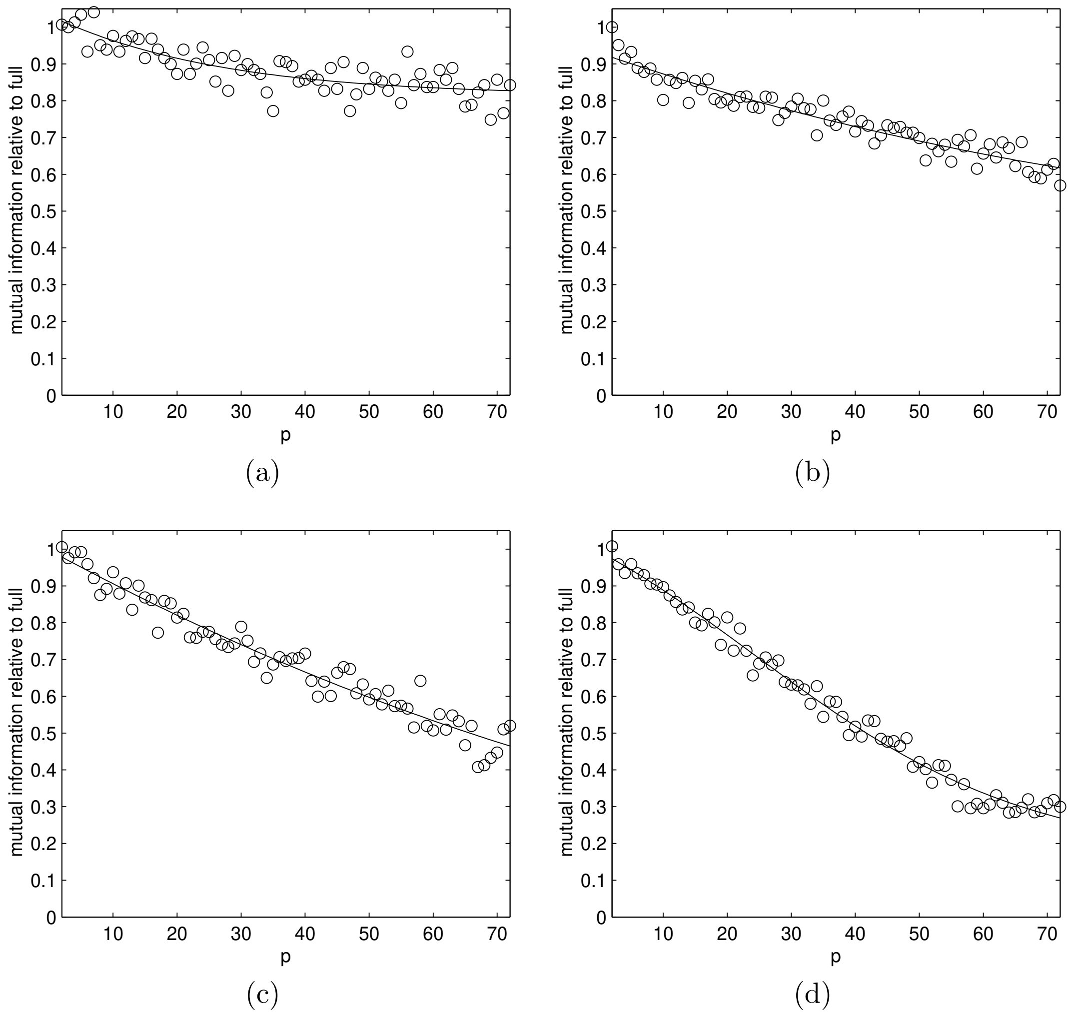


Figure 8: ODC Results. Clustering quality (measured by relative mutual information) as a function of the number of partitions, p, on various data-sets: (a) `2D2K`; (b) `8D5K`; (c) `PENDIG`; (d) `YAHOO`. The sum of the number of samples over all partitions is fixed at $2n$. Each plot contains experimental results using graph partitioning in the inner loop for $p = [2, \ldots, 72]$ and a fitted sigmoid. Processing time can be reduced by a factor of 4 for the `YAHOO` data while preserving 80% of the quality ($p = 16$).

and one uses only MCLA and HGPA in the supra-consensus function.[10] We define speedup as the computation time for the full clustering divided by the time when using the ODC approach. The overhead for the MCLA and HGPA consensus functions grows linearly in n and is negligible compared to the $O(n^2)$ clustering. Hence the asymptotic sequential speedup is approximately $s^{(\mathrm{ODC-SEQ})} \approx \frac{p}{v^2}$. Each partition can be clustered without any communication on a separate processor. At integration time only the n-dimensional label vector (instead of the entire $n \times n$ similarity matrix) has to be transmitted to the combiner. Hence, ODC does not only save computation time, but also enables trivial p-fold parallelization. Consequently, if a p-processor computer is utilized, an asymptotic speedup of $s^{(\mathrm{ODC-PAR})} \approx \frac{p^2}{v^2}$ is obtained. For example, when $p = 4$ and $v = 2$ the computing time is approximately the same because each partition is half the original size $n/2$ and consequently processed in a quarter of the time. Since there are four partitions, ODC takes the same time as the original processing. In our experiments, using partitions from 2 to 72 yields correspondingly approximate sequential (parallel) speedups from 0.5 (1) to 18 (1296). For example, `2D2K` (`YAHOO`) can be sped up 64-fold using 16 processors at 90% (80%) of the full length quality. In fact, `2D2K` can be clustered in less that 1/1000 of the original time at 80% quality.

4.5 Robust Centralized Clustering (RCC)

A consensus function can introduce redundancy and foster robustness when, instead of choosing or fine-tuning a single clusterer, an ensemble of clusterers is employed and their results are combined. This is particularly useful when clustering has to be performed in a closed loop without human interaction. The goal of robust centralized clustering (RCC) is to perform well for a wide variety of data distributions with a *fixed* ensemble of clusterers.

In RCC, each clusterer has access to all features and to all objects. However, each clusterer might take a different approach. In fact, approaches *should* be very diverse for best results. They can use different distance/similarity measures (Euclidean, cosine, etc.), or techniques (graph-based, agglomerative, k-means) (Strehl et al., 2000). The ensemble's clusterings are then integrated using the consensus function Γ without access to the original features.

To show that RCC can yield robust results in low-dimensional metric spaces as well as in high-dimensional sparse spaces *without* any modifications, the following experiment was set up. First, 10 diverse clustering algorithms were implemented: (1) self-organizing map; (2) hypergraph partitioning; k-means with distance based on (3) Euclidean, (4) cosine, (5) correlation, and (6) extended Jaccard; and graph partitioning with similarity based on (7) Euclidean, (8) cosine, (9) correlation, and (10) extended Jaccard. Implementation details of the individual algorithms can be found in Strehl et al. (2000).

RCC was performed 10 times each on sample sizes of 50, 100, 200, 400, and 800, for the `2DGA`, `8DGA`, and `PEND` data-sets. In case of the `YAHOO` data, only sample sizes of 200, 400, and 800 were used because fewer samples are not sufficient to meaningfully partition into 40 clusters. Different sample sizes provide insight into how cluster quality improves as more data becomes available. Quality improvement depends on the clusterer as well as the data. For example, more complex data-sets require more data until quality reaches

10. CSPA is $O(n^2)$ and would reduce speedups obtained by distribution.

a maximum. We also computed a random clustering for each experiment to establish a baseline performance. The random clustering consists of labels drawn from a uniform distribution from 1 to k. The quality in terms of difference in mutual information as compared to the random clustering algorithm for all 11 approaches (10 + consensus) is shown in Figure 9. Figure 10 shows learning curves for the average quality of the 10 algorithms versus RCC.

In Figure 9 (top row) the results for the `2D2K` data using $k = 2$ are shown. From an external viewpoint, the consensus function was given seven good clusterings (Euclidean, cosine, and extended Jaccard based k-means as well as graph partitioning, and self-organizing feature-map) and three poor clusterings (hypergraph partitioning, correlation based k-means, and correlation based graph partitioning). At sample size of $n = 800$, the RCC results are better than those for each individual algorithm quality evaluations. There is no noticeable deterioration caused by the poor clusterings. The average RCC mutual information-based quality $\phi^{(\mathrm{NMI})}$ of 0.85 is 48% higher than the average quality of all individual algorithms (excluding random) of 0.57.

In the case of the `YAHOO` data (Figure 9, bottom row) the consensus function received three poor clusterings (Euclidean based k-means as well as graph partitioning; and self-organizing feature-map, four good clusterings (hypergraph partitioning, cosine, correlation, and extended Jaccard based k-means) and three excellent clusterings (cosine, correlation, and extended Jaccard based graph partitioning). The RCC results are almost as good as the average of the excellent clusterers despite the presence of distractive clusterings. In fact, at the $n = 800$ level, RCC's average quality of 0.38 is 15% better than the average qualities of all the other algorithms (excluding random) at 0.33. This shows that for this scenario, too, cluster ensembles work well and also are robust!

Similar results are obtained for `8D5K` and `PENDIG`. In these two cases, all individual approaches work comparably well except for hypergraph-partitioning. The supra-consensus function learns to ignore hypergraph-partitioning results and yields a consensus clustering of good quality.

Figure 10 shows how RCC is consistently better in all four scenarios than picking a random/average single technique. Looking at the three consensus techniques, the need for all of them becomes apparent since there is no clear winner. In `2D2K`, `8D5K`, and `PENDIG`, MCLA generally had the highest ANMI, followed by CSPA, while HGPA performed poorly. In `YAHOO`, both CSPA and HGPA, had the highest ANMI approximately equally often, while MCLA performed poorly. We believe this is due to the fact that there was more diversity in `YAHOO` clusterings. CSPA and HGPA are better suited for that because no cluster correspondence is assumed.

The experimental results clearly show that cluster ensembles can be used to increase robustness in risk-intolerant settings. Since it is generally hard to evaluate clusters in high-dimensional problems, a cluster ensemble can be used to 'throw' many models at a problem and then integrate them using an consensus function to yield stable results. Thereby the user does not have to have category labels to pick a single best model. Rather, the ensemble automatically 'focuses' on whatever is most appropriate for the given data. In our experiments, there is diversity as well as some poorly performing clusterers. If there are diverse but comparably performing clusterers, the quality actually significantly outperforms the best individual clusterer, as seen in subsection 4.3.

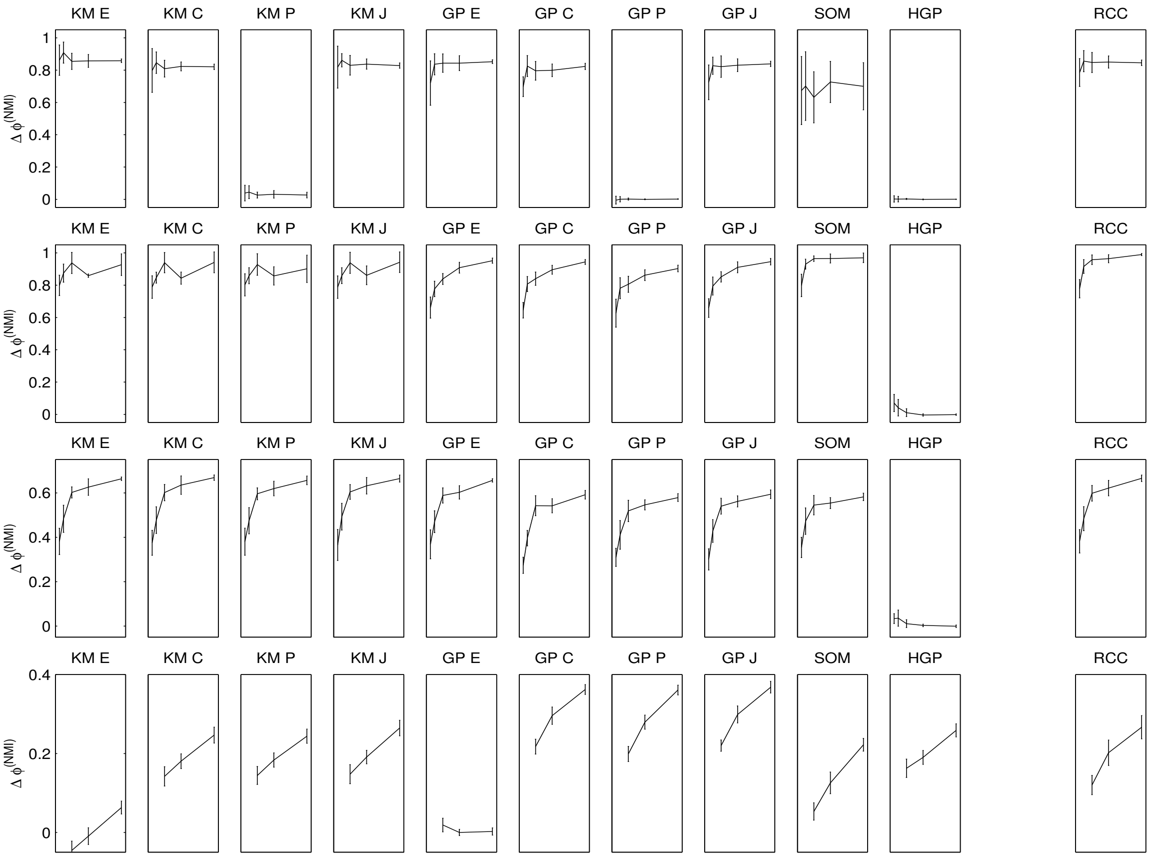


Figure 9: Detailed RCC results. Learning curves for `2D2K` (top row), `8D5K` (second row), `PENDIG` (third row), and `YAHOO` (bottom row) data. Each learning curve shows the difference in mutual information-based quality $\phi^{(\mathrm{NMI})}$ compared to random for five sample sizes at 50, 100, 200, 400, and 800. The bars for each data-point indicate ± 1 standard deviations over 10 experiments. Each column corresponds to a particular clustering algorithm. The rightmost column gives RCC quality for combining results of all 10 other algorithms. RCC yields robust results in all four scenarios.

5. Related Work

As mentioned in the introduction, there is an extensive body of work on combining multiple classifiers or regression models (Sharkey, 1996, Ghosh, 2002b), but relatively little to date on combining multiple clusterings in the machine learning literature. However, in traditional pattern recognition, there is a substantial body of largely theoretical work on *consensus classification* from the mid-80's and earlier (Neumann and Norton, 1986a,b, Barthelemy et al., 1986). These studies used the term 'classification' in a very general sense, encompassing partitions, dendrograms and n-trees as well. Today, such operations are typically referred to as clusterings. In consensus classification, a profile is a set of classifications

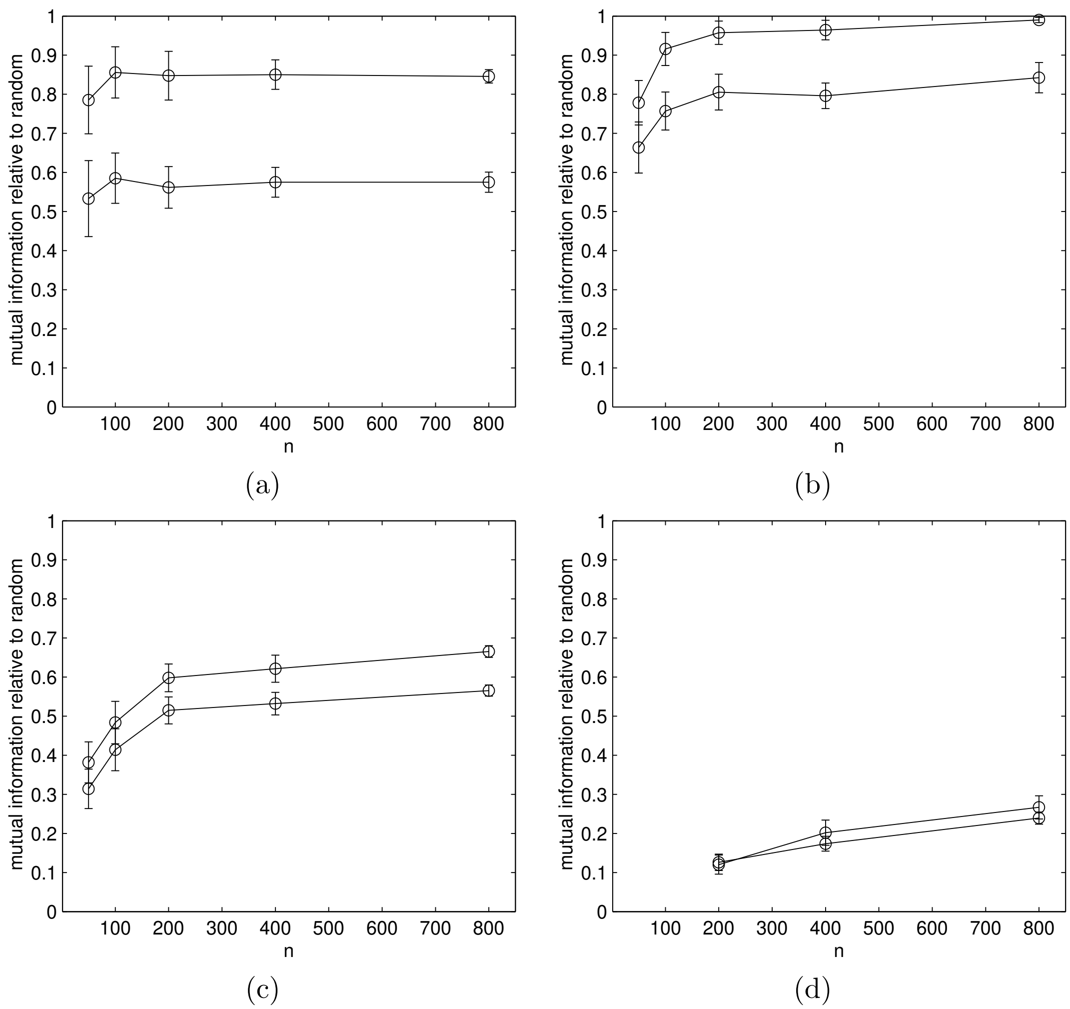


Figure 10: Summary of RCC results. Average learning curves and RCC learning curves for `2D2K` (a), `8D5K` (b), `PENDIG` (c), and `YAHOO` (d) data. Each learning curve shows the difference in mutual information-based quality $\phi^{(\mathrm{NMI})}$ compared to random for five sample sizes (at 50, 100, 200, 400, and 800). The bars for each datapoint indicate ± 1 standard deviations over 10 experiments. The upper curve gives RCC quality for combining results of all other 10 algorithms. The lower curve is the average performance of the 10 algorithms. RCC yields robust results in any scenario.

which is sought to be integrated into a single consensus classification. They were largely interested in obtaining a strict consensus, which identifies the supremum and the infimum of all given pairs of clusterings, using the relation 'sub-cluster' to define the partial ordering. Note that, unlike our approach, strict consensus results are not at the same level of scale or resolution as the original clusterings. In fact, in the presence of strong noise the results would often be trivial, that is, the supremum would be a single cluster of all objects, and the infimum returned the set of n singletons. Also, the techniques were computationally expensive and meant for smaller data-sets.

The most prominent application of strict consensus is by the computational biology community to obtain phylogenetic trees (Kim and Warnow, 1999, Kannan et al., 1995). A set of DNA sequences can be used to generate evolutionary trees using criteria such as maximum parsimony, but often one obtains several hundred trees with the same score function. In such cases, biologists look for the strict *consensus tree*, the supremum, which has lower resolution but is compatible with all the individual trees. Note that such systems are different from cluster ensembles in that (i) they are hierarchical clusterings, typically using *unrooted trees*, (ii) they have domain specific distance metrics (for example, Robinson-Foulds distance) and evaluation criteria such as parsimony, specificity and density, and (iii) they require *strict* consensus.

A recent interesting application of consensus clustering to help in supervised learning was to generate a single decision tree from N-fold cross-validated C4.5 results (Kavšek et al., 2001). Objects are first clustered according to their positions or paths in the N decision trees. Then, only objects of the majority class in each cluster are selected to form a new training set that generates the consensus decision tree. The goal here is to obtain a single, simplified decision tree without compromising much on classification accuracy.

There are several techniques where multiple clusterings are created and evaluated as intermediate steps in the process of attaining a single, higher quality clustering. For example, Fisher examined methods for iteratively improving an initial set of hierarchical clustering solutions (Fisher, 1996). Fayyad et al. (1998) presented a way of obtaining multiple approximate k-means solutions in main memory after making a single pass through a database, and then combining these means to get a final set of cluster centers. In all of these works, a summary representation of each cluster in terms of the base features is available to the integration mechanism, as opposed to our knowledge reuse framework, wherein only cluster labels are available. More recently, an 'evidence accumulation' framework was proposed wherein multiple k-means, using a much higher value of k than the final anticipated answer, were run on a common data-set (Fred and Jain, 2002). The results were used to form a co-occurrence or similarity matrix that records the fraction of solutions for which a given pair of points fell in the same cluster. Thus the effect of the multiple, fine-level clusterings is essentially to come up with a more robust similarity indicator having the flavor of the classical shared nearest-neighbors measure (Jarvis and Patrick, 1973). A single-link clustering is then used based on this similarity matrix. The co-occurrence matrix is analogous to the one used for CSPA, except, of course, that the clusterings generated in Fred and Jain (2002) are not legacy and are at a finer level of resolution.

One use of cluster ensembles is to exploit multiple existing groupings of the data. Several analogous approaches exist in supervised learning scenarios where class labels are known, under categories such as 'life-long learning' (Thrun, 1996), 'learning to learn' (Thrun and

Pratt, 1997) and 'knowledge reuse' (Bollacker and Ghosh, 1998, 1999). Several researchers have attempted to directly reuse the internal state information from classifiers under the belief that related classification tasks may benefit from common internal features. One approach to this idea is to use the weights of hidden layers in a multi-layer perceptron (MLP) classifier architecture to represent the knowledge to be shared among the multiple tasks being trained on simultaneously (Caruana, 1995). Pratt (1994) uses some of the trained weights from one MLP network to initialize weights in another MLP to be trained for a later, related task. In a related work, Silver and Mercer (1996) have developed a system consisting of *task* networks and an *experience* network. The experience network tries to learn the converged weights of related task networks in order to initialize weights of target task networks. Both of these weight initialization techniques resulted in improved learning speed. Besides weight reuse in MLP-type classifiers, other state reuse methods have been developed. One approach, developed by Thrun and O'Sullivan (1996), is based on a nearest neighbor classifier in which each of the dimensions of the input space is scaled to bring examples within a class closer together while pushing examples between different classes apart. The scaling vector derived for one classification task is then used in another, related task. We have previously proposed a knowledge reuse framework wherein the labels produced by old classifiers are used to improve the generalization performance of a new classifier for a different but related task (Bollacker and Ghosh, 1998). This improvement is facilitated by a supra-classifier that accesses only the outputs of the old and new classifiers, and does not need the training data that was used to create the old classifiers. Substantial gains are achieved when the training set size for the new problem is small, but can be compensated for by the extraction of information from the existing related solutions.

Another application of cluster ensembles is to combine multiple clusterings that were obtained based on only partial sets of features. This problem has been approached recently as a case of collective data mining (Kargupta et al., 1999). In Johnson and Kargupta (1999) a feasible approach to combining distributed agglomerative clusterings is introduced. First, each local site generates a dendrogram. The dendrograms are collected and pairwise similarities for all objects are created from them. The combined clustering is then derived from the similarities. In Kargupta et al. (2001), a distributed method of principal components analysis is introduced for clustering.

The usefulness of having multiple views of data for better clustering has been recognized by others as well. In multi-aspect clustering (Modha and Spangler, 2000), several similarity matrices are computed separately and then integrated using a weighting scheme. Also, Mehrotra (1999) has proposed a multi-viewpoint clustering, where several clusterings are used to semi-automatically structure rules in a knowledge base. The usefulness of having multiple clusterings of objects that capture relatively independent aspects of the information these objects convey about a target set of variables, was one of the motivations behind the multivariate extension of the information bottleneck principle (Friedman et al., 2001).

Much interest has also emerged in semi-supervised methods that form a bridge between classification and clustering by augmenting a limited training set of labeled data by a larger amount of unlabelled data. One powerful idea is to use *co-training* (Blum and Mitchell, 1998), whose success hinges on the presence of multiple 'redundantly sufficient' views of the data. For example, Muslea et al. (2001) introduced a multi-view algorithm including active sampling based on co-training. Nigam and Ghani (2000) investigated the effectiveness of co-training in semi-supervised settings.

Our proposed algorithms use *hypergraph* representations which have been extensively studied (Garey and Johnson, 1979). Hypergraphs have been previously used for (a single) high-dimensional clustering (Han et al., 1997, Strehl et al., 2000), but not for combining multiple groupings. *Mutual information* (Cover and Thomas, 1991) is a useful measure in a variety of contexts. For example, the information bottleneck method (Slonim and Tishby, 2000) uses mutual information to reduce dimensionality while preserving as much information as possible about the class labels. Mutual information has also been used to evaluate clusterings by comparing cluster labels with class labels (Strehl et al., 2000), but not to integrate multiple clusterings.

Finally, at a conceptual level, the consensus functions operate on the similarities between pairs of objects, as indicated by their cluster labels. The attractiveness of considering objects embedded in a derived (dis)similarity space, as opposed to the original feature space, has been recently shown in several classification applications (Pekalska et al., 2002, Strehl and Ghosh, 2002b).

6. Concluding Remarks

In this paper we introduced the cluster ensemble problem and provided three effective and efficient algorithms to solve it. We defined a mutual information-based objective function that enables us to automatically select the best solution from several algorithms and to build a supra-consensus function as well. We conducted experiments to show how cluster ensembles can be used to introduce robustness, speedup superlinear clustering algorithms, and dramatically improve 'sets of subspace clusterings' for several quite different domains. In document clustering of Yahoo! web pages, we showed that combining, for example, 20 clusterings, each obtained from only 128 random words, can more than double quality compared to the best single result. Some of the algorithms and data-sets are available for download at `http://strehl.com/`.

The cluster ensemble is a very general framework that enables a wide range of applications. The purpose of this paper was to present the basic problem formulation and explore some application scenarios. There are several issues and aspects, both theoretical and practical, that are worthy of further investigation. For example, while the formulation allows different clusterers to provide varying numbers of clusters (k), in most of the experiments, k was the same for each clusterer. In reality, when clusterings are done in a distributed fashion, perhaps by different organizations with different data-views or goals, it is quite likely that k will vary from site to site. Is our consensus clustering fairly robust to such variations? Our preliminary results on this issue are reported in Ghosh et al. (2002). They indicate that cluster ensembles are indeed very helpful in determining a reasonable value of k for the consensus solution since the ANMI value peaks around the desirable range. Moreover, it typically gives better results than the best individual solution even when the ensemble members cluster the data at highly varying resolutions. This paper also provides further evidence of the suitability of the ANMI criterion as an indicator of NMI with the "true labels", since a plot of ANMI vs. NMI over a large number of experiments shows a correlation coefficient of about 0.93.

Other worthwhile future work includes a thorough theoretical analysis of the average normalized mutual information (ANMI) objective, including how it can be applied to soft clusterings. We also plan to explore possible greedy optimization schemes in more detail. The greedy scheme introduced in Section 2.3 is not very practical by itself. However, it can be used as a post-processing step to refine good solutions when n is not too large. For example, one can use the supra-consensus labeling as the initialization instead of the best single input clustering. Preliminary experiments indicate that this post-processing affects between 0% - 5% of the labels and yields slightly improved results. Another direction of future work is to better understand the biases of the three proposed consensus functions. We would also like to extend our application scenarios. In real applications, a variety of hybrids of the investigated FDC, ODC and RCC scenarios can be encountered. Cluster ensembles could enable federated data mining systems to work on top of distributed and heterogeneous databases.

Acknowledgments

We would like to acknowledge support from the NSF under Grant ECS-9900353, from a Faculty Partnership Award from IBM/Tivoli and IBM ACAS, and from Intel. We thank Claire Cardie for careful and prompt editing of this paper, and the anonymous referees for helpful comments.

References

C. J. Alpert and A. B. Kahng. Recent directions in netlist partitioning: A survey. *Integration: The VLSI Journal*, 19:1–18, 1995.

J.A. Barnett. Computational methods for a mathematical theory of evidence. In *Proc. of IJCAI*, pages 868–875, 1981.

J. P. Barthelemy, B. Laclerc, and B. Monjardet. On the use of ordered sets in problems of comparison and consensus of classifications. *Journal of Classification*, 3:225–256, 1986.

A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the 11th Annual Conference on Computational Learning Theory (COLT-98)*, pages 92–100, 1998.

D. Boley, M. Gini, R. Gross, E. Han, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore. Partitioning-based clustering for web document categorization. *Decision Support Systems*, 27:329–341, 1999.

Kurt D. Bollacker and Joydeep Ghosh. A supra-classifier architecture for scalable knowledge reuse. In *Proc. Int'l Conf. on Machine Learning (ICML-98)*, pages 64–72, July 1998.

Kurt D. Bollacker and Joydeep Ghosh. Effective supra-classifiers for knowledge base construction. *Pattern Recognition Letters*, 20(11-13):1347–52, November 1999.

P.S. Bradley and U. M. Fayyad. Refining initial points for K--means clustering. In *Proc. Int'l Conf. on Machine Learning (ICML-98)*, pages 91–99, July 1998.

Rich Caruana. Learning many related tasks at the same time with backpropagation. In *Advances in Neural Information Processing Systems 7*, pages 657–664, 1995.

S. V. Chakaravathy and J. Ghosh. Scale based clustering using a radial basis function network. *IEEE Transactions on Neural Networks*, 2(5):1250–61, Sept 1996.

Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley, 1991.

B. Dasarathy. *Decision Fusion*. IEEE CS Press, Los Alamitos, CA, 1994.

I. S. Dhillon and D. S. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1):143–175, January 2001.

T. G. Dietterich. Ensemble methods in machine learning. In J. Kittler and F. Roli, editors, *Multiple Classifier Systems*, pages 1–15. LNCS Vol. 1857, Springer, 2001.

M. Ester, H. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of 2nd International Conference on KDD*, pages 226–231, 1996.

U. M. Fayyad, C. Reina, and P. S. Bradley. Initialization of iterative refinement clustering algorithms. In *Proc. 14th Intl. Conf. on Machine Learning (ICML)*, pages 194–198, 1998.

Doug Fisher. Iterative optimization and simplification of hierarchical clusterings. *Journal of Artificial Intelligence Research*, 4:147–180, 1996.

W. Frakes. Stemming algorithms. In W. Frakes and R. Baeza-Yates, editors, *Information Retrieval: Data Structures and Algorithms*, pages 131–160. Prentice Hall, New Jersey, 1992.

A. L. N. Fred and A. K. Jain. Data clustering using evidence accumulation. In *Proc. ICPR*, page to appear, 2002.

N. Friedman, O. Mosenzon, N. Slonim, and N. Tishby. Multivariate information bottleneck. In *Proc. of the Seventeenth Conf. on Uncertainty in Artificial Intelligence (UAI)*. AAAI Press, 2001.

Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W. H. Freeman, San Francisco, CA, 1979.

J. Ghosh. Multiclassifier systems: Back to the future. Keynote Talk, 3rd Int'l Workshop on Multiple Classifier Systems, Cagliari, June, 2002a. Downloadable from http://www.lans.ece.utexas.edu/publications.html.

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

J. Ghosh, A. Strehl, and S. Merugu. A consensus framework for integrating distributed clusterings under limited knowledge sharing. In *Proc. NSF Workshop on Next Generation Data Mining, Baltimore*, pages 99–108, Nov 2002.

C. W. J. Granger. Combining forecasts–twenty years later. *Journal of Forecasting*, 8(3): 167–173, 1989.

E. Han, G. Karypis, V. Kumar, and B. Mobasher. Clustering in a highdimensional space using hypergraph models. Technical Report 97-019, University of Minnesota, Department of Computer Science, 1997.

A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, New Jersey, 1988.

R. A. Jarvis and E. A. Patrick. Clustering using a similarity measure based on shared nearest neighbors. *IEEE Transactions on Computers*, C-22, No. 11:1025–1034, 1973.

E. Johnson and H. Kargupta. Collective, hierarchical clustering from distributed, heterogeneous data. In M. Zaki and C. Ho, editors, *Large-Scale Parallel KDD Systems*, volume 1759 of *Lecture Notes in Computer Science*, pages 221–244. Springer-Verlag, 1999.

S. Kannan, T. Warnow, and S. Yooseph. Computing the local consensus of trees. In *Association for Computing Machinery and the Society of Industrial Applied Mathematics, Proceedings, ACM/SIAM Symposium on Discrete Algorithms*, pages 68–77, 1995.

H. Kargupta and P. Chan, editors. *Advances in Distributed and Parallel Knowledge Discovery.* AAAI/MIT Press, Cambridge, MA, 2000.

H. Kargupta, W. Huang, Krishnamoorthy, and E. Johnson. Distributed clustering using collective principal component analysis. *Knowledge and Information Systems Journal Special Issue on Distributed and Parallel Knowledge Discovery*, 3:422–448, 2001.

H. Kargupta, B. Park, D. Hershberger, and E. Johnson. Collective data mining: A new perspective toward distributed data mining. In Hillol Kargupta and Philip Chan, editors, *Advances in Distributed and Parallel Knowledge Discovery.* MIT/AAAI Press, 1999.

G. Karypis, E.-H. Han, and V. Kumar. Chameleon: Hierarchical clustering using dynamic modeling. *IEEE Computer*, 32(8):68–75, August 1999.

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, 1998.

George Karypis, Rajat Aggarwal, Vipin Kumar, and Shashi Shekhar. Multilevel hypergraph partitioning: Applications in VLSI domain. In *Proceedings of the Design and Automation Conference*, 1997.

Branko Kavšek, Nada Lavrač, and Anuška Ferligoj. Consensus decision trees: Using consensus hierarchical clustering for data relabelling and reduction. In *Proceedings of ECML 2001*, volume 2167 of *LNAI*, pages 251–262. Springer, 2001.

B. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell Systems Technical Journal*, 49:291–307, 1970.

J. Kim and T. Warnow. Tutorial on phylogenetic tree estimation. In *Intelligent Systems for Molecular Biology, Heidelberg*, 1999.

J. Kittler and F. Roli, editors. *Multiple Classifier Systems.* LNCS Vol. 2634, Springer, 2002.

Teuvo Kohonen. *Self-Organizing Maps.* Springer, Berlin, Heidelberg, 1995. (Second Extended Edition 1997).

A. Krogh and J. Vedelsby. Neural network ensembles, cross validation and active learning. In D.S. Touretzky G. Tesauro and T.K. Leen, editors, *Advances in Neural Information Processing Systems-7*, pages 231–238, 1995.

Mala Mehrotra. Multi-viewpoint clustering analysis (mvp-ca) technology for mission rule set development and case-based retrieval. Technical Report AFRL-VS-TR-1999-1029, Air Force Research Laboratory, 1999.

Dharmendra S. Modha and W. Scott Spangler. Clustering hypertext with applications to web searching. In *Proceedings of the ACM Hypertext 2000 Conference, San Antonio, TX, May 30-June 3*, 2000.

Ion Muslea, Steve Minton, and Craig Knoblock. Selective sampling + semi-supervised learning = robust multi-view learning. In *IJCAI-2001 Workshop on Text Learning Beyond Supervision*, 2001.

D. A. Neumann and V. T. Norton. Clustering and isolation in the consensus problem for partitions. *Journal of Classification*, 3:281–298, 1986a.

D. A. Neumann and V. T. Norton. On lattice consensus methods. *Journal of Classification*, 3:225–256, 1986b.

K. Nigam and R. Ghani. Analyzing the applicability and effectiveness of co-training. In *Proceedings of CIKM-00, 9th ACM International Conference on Information and Knowledge Management*, pages 86–93. ACM, 2000.

E. Pekalska, P. Paclik, and R.P.W. Duin. A generalized kernel approach to dissimilarity-based classification. *Journal of Machine Learning Research, Special Issue on Kernel Methods*, 2(2):175–211, 2002.

Lorien Y. Pratt. Experiments on the transfer of knowledge between neural networks. In S. Hanson, G. Drastal, and R. Rivest, editors, *Computational Learning Theory and Natural Learning Systems, Constraints and Prospects*, chapter 19, pages 523–560. MIT Press, 1994.

A. Prodromidis, P. Chan, and S. Stolfo. Meta-learning in distributed data mining systems: Issues and approaches. In H. Kargupta and P. Chan, editors, *Advances in Distributed and Parallel Knowledge Discovery.* AAAI/MIT Press, Cambridge, MA, 2000.

M.D. Richard and R.P. Lippmann. Neural network classifiers estimate bayesian a posteriori probabilities. *Neural Computation*, 3(4):461–483, 1991.

A. Sharkey. On combining artificial neural networks. *Connection Science*, 8(3/4):299–314, 1996.

A. Sharkey. *Combining Artificial Neural Nets*. Springer-Verlag, 1999.

D. Silver and R. Mercer. The parallel transfer of task knowledge using dynamic learning rates based on a measure of relatedness. *Connection Science Special Issue: Transfer in Inductive Systems*, 1996.

N. Slonim and N. Tishby. Agglomerative information bottleneck. In *Proc. of NIPS-12*, pages 617–623. MIT Press, 2000.

A. Strehl and J. Ghosh. Cluster ensembles – a knowledge reuse framework for combining partitionings. In *Proceedings of AAAI 2002, Edmonton, Canada*, pages 93–98. AAAI, July 2002a.

Alexander Strehl and Joydeep Ghosh. A scalable approach to balanced, high-dimensional clustering of market-baskets. In *Proc. HiPC 2000, Bangalore*, volume 1970 of *LNCS*, pages 525–536. Springer, December 2000.

Alexander Strehl and Joydeep Ghosh. Relationship-based clustering and visualization for high-dimensional data mining. *INFORMS Journal on Computing*, 2002b. in press.

Alexander Strehl, Joydeep Ghosh, and Raymond Mooney. Impact of similarity measures on web-page clustering. In *Proc. AAAI Workshop on AI for Web Search (AAAI 2000), Austin*, pages 58–64. AAAI, July 2000.

S. Thrun. Is learning the n-th thing any easier than learning the first? In M.C. Mozer D.S. Touretzky and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems-8*, pages 640–646. MIT Press, Cambridge, MA, 1996.

S. Thrun and L.Y. Pratt. *Learning To Learn*. Kluwer Academic, Norwell, MA, 1997.

Sebastian Thrun and Joseph O'Sullivan. Discovering structure in multiple learning tasks: The TC alogorithm. In *The 13th International Conference on Machine Learning*, pages 489–497, 1996.

K. Tumer and J. Ghosh. Linear and order statistics combiners for pattern classification. In A. Sharkey, editor, *Combining Artificial Neural Nets*, pages 127–162. Springer-Verlag, 1999.