# Cluster Ensembles – A Knowledge Reuse Framework for Combining Partitionings

**Alexander Strehl** and **Joydeep Ghosh**[*]
Department of Electrical and Computer Engineering
The University of Texas at Austin
Austin, TX 78712, USA
strehl@ece.utexas.edu and ghosh@ece.utexas.edu

## Abstract

It is widely recognized that combining multiple classification or regression models typically provides superior results compared to using a single, well-tuned model. However, there are no well known approaches to combining multiple non-hierarchical clusterings. The idea of combining cluster labelings without accessing the original features leads us to a general knowledge reuse framework that we call *cluster ensembles*. Our contribution in this paper is to formally define the cluster ensemble problem as an optimization problem and to propose three effective and efficient combiners for solving it based on a hypergraph model. Results on synthetic as well as real data sets are given to show that cluster ensembles can (i) improve quality and robustness, and (ii) enable distributed clustering.

## 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, evidences in rule-based systems and multi-sensor data fusion. A simple but effective type of such multi-learner systems are *ensembles*, wherein each component learner (typically a regressor or classifier) tries to solve the same task (Sharkey 1996; Tumer & Ghosh 1996).

But unlike classification and regression problems, there are no well known approaches to combining multiple clusterings which 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 presented to that method.

A *clusterer* consists of a particular clustering algorithm with a specific view of the data. A *clustering* is the output of a clusterer and consists of the group labels for some or all objects. *Cluster ensembles* provide a tool for consolidation of results from a portfolio of individual clustering results. It is useful in a variety of contexts:

- *Quality and Robustness*. Combining several clusterings can lead to *improved quality and robustness* of results. Diversity among individual clusterers can be achieved by using different features to represent the objects, by 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, or by using a portfolio of diverse clustering algorithms.
- *Knowledge Reuse* (Bollacker & Ghosh 1998). Another important consideration is the *reuse of existing clusterings*. In several applications, a variety of clusterings for the objects under consideration may already exist. For example, in grouping customers for a direct marketing campaign, various legacy customer segmentations might already exist, based on demographics, credit rating, geographical region, or income tax bracket. In addition, customers can be clustered based on their purchasing history (e.g., market-baskets) (Strehl & Ghosh 2002). Can we reuse such pre-existing knowledge to create a single consolidated clustering? Knowledge reuse in this context means that we exploit the information in the provided cluster labels *without* going back to the *original features* or the algorithms that were used to create the clusterings.
- *Distributed Computing*. The ability to deal with clustering in a distributed fashion is useful to improve scalability, security, and reliability. Also, real applications often involve distributed databases due to organizational or operational constraints. A cluster ensemble can combine individual results from the distributed computing entities.

**Related Work**. As mentioned in the introduction, there is an extensive body of work on combining multiple classifiers or regression models, but little on combining multiple clusterings so far in the machine learning community. However, there was a substantial body of largely theoretical work on *consensus classification* during the mid-80s and earlier (Barthelemy, Laclerc, & Monjardet 1986). These studies used the term 'classification' in a very general sense, encompassing partitions, dendrograms and $n$-trees as well. In consensus classification, a profile is a set of classifications which is sought to be integrated into a single consensus classification. A representative work is that of (Neumann & Norton 1986), who investigated techniques for strict consensus. Their approach is based on the construction of a lattice over the set of all partitionings by using a refinement relation.

[*]This research was supported in part by the NSF under Grant ECS-9900353, by IBM and Intel.

Such work on strict consensus works well for small data-sets with little noise and little diversity and obtains solution on a *different* level of resolution. The most prominent application of strict consensus is for the computational biology community to obtain phylogenetic trees (Kim & Warnow 1999).

One use of cluster ensembles proposed in this paper is to exploit multiple existing groupings of the data. Several analogous approaches exist in supervised learning scenarios (class labels are known), under categories such as 'life-long learning', 'learning to learn', and 'knowledge reuse' (Bollacker & Ghosh 1999). Several representative works can be found in (Thrun & Pratt 1997).

Another application of cluster ensembles is to combine multiple clustering that were obtained based on only partial sets of features. This problem has been approached recently as a case of collective data mining. In (Johnson & 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.

The usefulness of having multiple views of data for better clustering has been recognized by others as well. In multi-aspect clustering (Modha & Spangler 2000), several similarity matrices are computed separately and then integrated using a weighting scheme. Also, Mehrotra has proposed a multi-viewpoint clustering, where several clusterings are used to semi-automatically structure rules in a knowledge base (Mehrotra 1999).

**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 $\lambda \in \mathbb{N}^n$. We also refer to a clustering/labeling as a model. A clusterer $\Phi$ is a function that delivers a label vector given an ordered set of objects. We call the original set of labelings, the *profile* and the resulting single labeling, the *consensus clustering*. Vector/matrix transposition is indicated with a superscript $\dagger$.

## The Cluster Ensemble Problem

### Illustrative Example

First, we will illustrate combining of clusterings using a simple example. Let the label vectors as given on the left of table 1 specify four clusterings. Inspection of the label vectors reveals that clusterings 1 and 2 are logically identical. Clustering 3 introduces some dispute about objects 3 and 5. Clustering 4 is quite inconsistent with all the other ones, has two groupings instead of 3, and also contains missing data. Now let us look for a good combined clustering with 3 clusters. Intuitively, a good combined clustering should *share as much information as possible* with the given 4 labelings. Inspection suggests that a reasonable integrated clustering is $(2, 2, 2, 3, 3, 1, 1)^\dagger$ (or one of the 6 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 7 elements into 3 groups, it can be shown that this clustering shares the maximum information with the given 4 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 general, the number of clusters as well as each cluster's interpretation may vary tremendously among models.

### Objective Function for Cluster Ensembles

Given $r$ groupings with the $q$-th grouping $\lambda^{(q)}$ having $k^{(q)}$ clusters, a consensus function $\Gamma$ is defined as a function $\mathbb{N}^{n \times r} \rightarrow \mathbb{N}^n$ mapping a set of clusterings to an integrated clustering: $\Gamma : \{\lambda^{(q)} \mid q \in \{1, \ldots, r\}\} \rightarrow \lambda$. The optimal combined clustering should share the most information with the original clusterings. But how do we measure shared information between clusterings? In information theory, mutual information is a symmetric measure to quantify the statistical information shared between two distributions. Suppose there are two labelings $\lambda^{(a)}$ and $\lambda^{(b)}$. Let there be $k^{(a)}$ groups in $\lambda^{(a)}$ and $k^{(b)}$ groups in $\lambda^{(b)}$. Let $n^{(h)}$ be the number of objects in cluster $\mathcal{C}_h$ according to $\lambda^{(a)}$, and $n_\ell$ the number of objects in cluster $\mathcal{C}_\ell$ according to $\lambda^{(b)}$. Let $n_\ell^{(h)}$ denote the number of objects that are in cluster $h$ according to $\lambda^{(a)}$ as well as in group $\ell$ according to $\lambda^{(b)}$. Then, a [0,1]-normalized mutual information criterion $\phi^{(\mathrm{NMI})}$ is computed as follows (Strehl, Ghosh, & Mooney 2000):

$$\phi^{(\mathrm{NMI})}(\lambda^{(a)}, \lambda^{(b)}) = \frac{2}{n} \sum_{\ell=1}^{k^{(a)}} \sum_{h=1}^{k^{(b)}} n_\ell^{(h)} \log_{k^{(a)} \cdot k^{(b)}} \left( \frac{n_\ell^{(h)} n}{n^{(h)} n_\ell} \right) \tag{1}$$

We propose that the optimal combined clustering $\lambda^{(k-\mathrm{opt})}$ be defined as the one that has maximal average mutual information with all individual labelings $\lambda^{(q)}$ given that the number of consensus clusters desired is $k$. Thus, our objective function is Average Normalized Mutual Information (ANMI). Then, $\lambda^{(k-\mathrm{opt})}$ is defined as

$$\lambda^{(k-\mathrm{opt})} = \arg\max_{\hat{\lambda}} \sum_{q=1}^{r} \phi^{(\mathrm{NMI})}(\hat{\lambda}, \lambda^{(q)}) \tag{2}$$

where $\hat{\lambda}$ goes through all possible $k$-partitions. The sum in equation 2 represents $r$ANMI. For finite populations, the trivial solution is to exhaustively search through all possible clusterings with $k$ labels (approximately $k^n/k!$ for $n \gg k$) for the one with the maximum ANMI which is computationally prohibitive. If some labels are missing one can still apply equation 2 by summing only over the object indices with known labels.

## 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.

### Representing Sets of Clusterings as a Hypergraph

The first step for 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.

|  | $\lambda^{(1)}$ | $\lambda^{(2)}$ | $\lambda^{(3)}$ | $\lambda^{(4)}$ |
|---|---|---|---|---|
| $x_1$ | 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 | ? |

⇔

|  | $\mathbf{H}^{(1)}$ |  |  | $\mathbf{H}^{(2)}$ |  |  | $\mathbf{H}^{(3)}$ |  |  | $\mathbf{H}^{(4)}$ |  |
|---|---|---|---|---|---|---|---|---|---|---|---|
|  | $\mathbf{h}_1$ | $\mathbf{h}_2$ | $\mathbf{h}_3$ | $\mathbf{h}_4$ | $\mathbf{h}_5$ | $\mathbf{h}_6$ | $\mathbf{h}_7$ | $\mathbf{h}_8$ | $\mathbf{h}_9$ | $\mathbf{h}_{10}$ | $\mathbf{h}_{11}$ |
| $v_1$ | 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.

For each label vector $\lambda^{(q)} \in \mathbb{N}^n$, we construct the binary membership indicator matrix $\mathbf{H}^{(q)} \in \mathbb{N}^{n \times k^{(q)}}$ in which each cluster is represented as a hyperedge (column), as illustrated in table 1. 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 $\mathbf{H} = (\mathbf{H}^{(1)} \ldots \mathbf{H}^{(r)})$ 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.

## Cluster-based Similarity Partitioning Algorithm (CSPA)

Essentially, if two objects are in the same cluster then they are considered to be fully similar, and if not they are fully dissimilar. This is the simplest heuristic and is used in the Cluster-based Similarity Partitioning Algorithm (CSPA) So, one can define similarity as the fraction of clusterings in which two objects are in the same cluster. The entire $n \times n$ similarity matrix $\mathbf{S}$ can be computed in one sparse matrix multiplication $\mathbf{S} = \frac{1}{r}\mathbf{H}\mathbf{H}^{\dagger}$. Now, we can use $\mathbf{S}$ to recluster the objects using any reasonable similarity based clustering algorithm. We chose to partition the induced graph (vertex = object, edge weight = similarity) using METIS (Karypis & Kumar 1998) because of its robust and scalable properties.

## HyperGraph-Partitioning Algorithm (HGPA)

The second algorithm is another 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 hyper-edges. We call this approach the HyperGraph-Partitioning Algorithm (HGPA). All hyperedges are considered to have the same weight. Also, all vertices are equally weighted. Now, we look for a hyperedge separator that partitions the hypergraph into $k$ unconnected components of approximately the same size. Equal sizes are obtained by maintaining a vertex imbalance of at most 5% as formulated by 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. We use the hypergraph partitioning package HMETIS (Karypis *et al.* 1997), which gives high-quality partitions and is very scalable.

## 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 represented 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 a regular undirected graph, the meta-graph. The edge 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 $\mathbf{h}_b$: $w_{a,b} = \frac{\mathbf{h}_a^{\dagger}\mathbf{h}_b}{\|\mathbf{h}_a\|_2^2 + \|\mathbf{h}_b\|_2^2 - \mathbf{h}_a^{\dagger}\mathbf{h}_b}$. Since the clusters are non-overlapping (e.g., hard), there are no edges amongst vertices of the same clustering $\mathbf{H}^{(q)}$ and, thus, the meta-graph is $r$-partite.
- **Cluster Hyperedges**. Find matching labels by partitioning the meta-graph into $k$ balanced meta-clusters. This results in a clustering of the $\mathbf{h}$ vectors in $\mathbf{H}$. 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. 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 $\lambda$.

In this paper, for all experiments *all three* algorithms, CSPA, HGPA and MCLA, are run and the clustering with the greater ANMI is used. Note that this procedure is still completely unsupervised. We call this the **supra-consensus function**.

## Applications and Results

Consensus functions *enable* a variety of new approaches to several problems. After introducing the data-sets and evaluation methodology used, we illustrate how robustness can be increased and how distributed clustering clustering can be performed when each entity only has access to a limited subset of features.

**Data-sets**. We illustrate the cluster ensemble applications on two real and two artificial data-sets. The first data-set (`2D2K`) was artificially generated and contains 500 points each of two 2-dimensional (2D) Gaussian clusters. The second data-set (`8D5K`) contains 1000 points from 5 multivariate Gaussian distributions (200 points each) in 8D. Both are available from `http://strehl.com/`. The third data-set (`PENDIG`) is for pen-based recognition of handwritten digits from 16 spatial features (available from UCI Machine Learning Repository). There are ten classes of roughly equal size corresponding to the digits 0 to 9. The fourth data-set (`YAHOO`) is for news text clustering (available from `ftp://ftp.cs.umn.edu/dept/users/boley/`). The raw $21839 \times 2340$ word-document matrix consists of the non-normalized occurrence frequencies of stemmed words, using Porter's suffix stripping algorithm. Pruning all words that occur less than $0.01$ or more than $0.10$ times on average because they are insignificant or too generic, results in 2903. For `2D2K`, `8D5K`, `PENDIG`, and `YAHOO` we use $k =$ 2, 5, and 10, 40 respectively.

**Quality Evaluation Criterion**. Standard cost functions, such as the sum of squared distances from the cluster representative, depend on metric interpretations of the data. They cannot be used to compare techniques that use different distance/similarity measures or work entirely without them. However, when objects have already been categorized by an external source, i.e. when class labels are available, we can use information theoretic measures to indicate the match between cluster labels and class labels. Previously, average purity and entropy-based measures to assess clustering quality from 0 (worst) to 1 (best) have been used. While average purity is intuitive to understand, it is biased to favor small clusters (singletons, in fact, are scored as perfect). Also, a monolithic clustering (one single cluster for all objects) receives a score as high as the maximum category prior probability. Using a measure based on normalized mutual information (equation 1) fixes both biases (Strehl, Ghosh, & Mooney 2000) and provides an unbiased and conservative measure as compared to purity and entropy. Given $g$ categories (or classes), we use the categorization labels $\kappa$ to evaluate the quality by normalized mutual information $\phi^{(\mathrm{NMI})}(\kappa, \lambda)$, as defined in equation 1.

### 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.

In Robust Centralized Clustering (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 similarity/distance measures (e.g., Euclidean, cosine) or techniques (graph-based, agglomerative, $k$-means) (Strehl, Ghosh, & Mooney 2000). The ensemble's clusterings are then integrated using the supra-consensus function.

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, (6) extended Jaccard; and graph partitioning with similarity based on (7) Euclidean, (8) cosine, (9) correlation, (10) extended Jaccard. Implementational details of the individual algorithms are beyond the scope of this paper and can be found in (Strehl, Ghosh, & Mooney 2000).

RCC was performed 10 times each on sample sizes of 50, 100, 200, 400, and 800, for each data-set. Different sample sizes provide insight 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 maxes out. We also computed a random clustering for each experiment to establish a baseline performance. The quality in terms of difference in mutual information as compared to the random clustering algorithm is computed for all 11 approaches (10 + consensus). Figure 1 shows learning curves for the average quality of the 10 algorithms versus RCC.

E.g., in case of the `YAHOO` data (figure 1(d)) the consensus function received three poor clusterings (Euclidean based $k$-means as well as graph partitioning; and self-organizing feature-map), four good (hypergraph partitioning, cosine, correlation, and extended Jaccard based $k$-means) and three excellent (cosine, correlation, and extended Jaccard based graph partitioning) clusterings. 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 19% better than the average qualities of all the other algorithms (excluding random) at 0.32. This shows that for this scenario, cluster ensembles work well and also are robust!

Similar results are obtained for 2D2K, 8D5K and PENDIG. Figure 1 shows how RCC is consistently better in all four scenarios than picking a random / average single technique. The experimental results clearly show that cluster ensembles can be used to increase robustness in risk intolerant settings. Especially, 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.

## Feature-Distributed Clustering (FDC)

In Feature-Distributed Clustering (FDC), we show how cluster ensembles can be used to boost quality of results by combining a set of clusterings obtained from partial views of the data. 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 access to all objects. The clusterers find groups in their views/subspaces. In the combining stage, individual results are integrated to recover the full structure of the data (without access to any of the original features).

First, let us discuss results for the 8D5K data, since they lend themselves well to illustration. We create 5 random 2D views (through selection of a pair of features) of the 8D data, and use Euclidean-based similarity and graph-partitioning with $k = 5$ in each view to obtain 5 clusterings. Clustering in the 8D space yields the original generative labels and is referred to as the reference clustering. Using PCA to project the data into 2D separates all 5 clusters fairly well (figure 3). In figure 3(a), the reference clustering is illustrated by coloring the data points in the space spanned by the first and second principal components (PCs). Figure 3(b) shows the final FDC result after combining 5 subspace clusterings. Each clustering has been computed from a random feature pair. These subspaces are shown in figure 2. Each of the rows corresponds to a random selection of 2 out of 8 feature dimensions. 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 amongst all plots. FDC (figure 3(b)) are clearly superior compared to any of the 5 individual results (figure 2(right)) and is almost flawless compared to the reference clustering (figure 3(a)).

We also conducted experiments on the other four datasets. Table 2 summarizes the results. The choice of the number of random 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 amongst the results was 0.17 and the best quality was 0.21. Using the supra-consensus function to combine all 20 labelings results in a quality of 0.38, or 124% higher mutual information than the average individual clustering. In all scenarios, the consensus clustering is as good or better than the best individual input clustering and always better than the average quality of individual clusterings. When

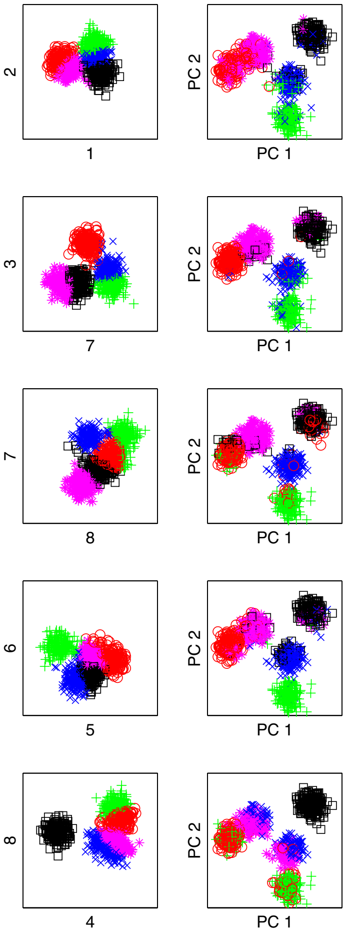


Figure 2: Illustration of feature distributed clustering (FDC) on 8D5K data. Each row corresponds to a random selection of 2 out of 8 feature dimensions. For each of the 5 chosen feature pairs, a row shows the clustering (colored) obtained on the 2D subspace spanned by the selected feature pair (left) and visualizes these clusters on the plane of the global 2 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 5 clusterings into a far superior clustering (figure 3(b)).

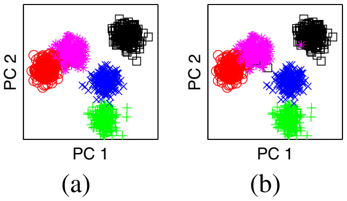
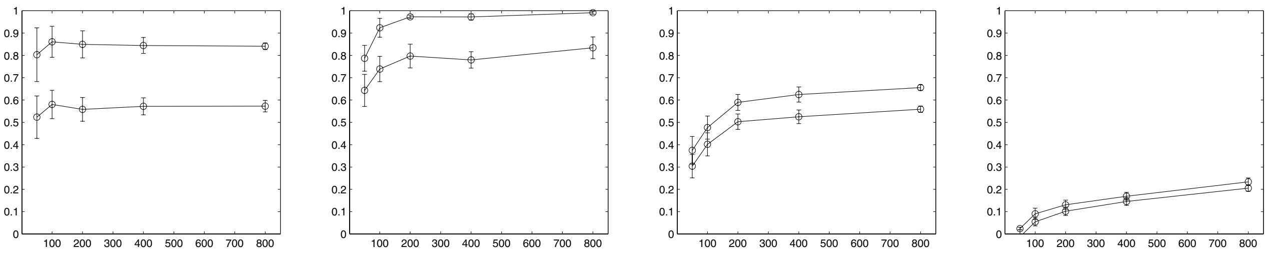


Figure 3: Reference clustering (a) and FDC consensus clustering (b) of 8D5K data shown in the first 2 principal components. The consensus clustering (b) is clearly superior compared to any of the 5 individual results (figure 2(right)) and is almost flawless compared to the reference clustering (a).



(a) (b) (c) (d)

Figure 1: 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 5 sample sizes. The bars for each data-point indicate ±1 standard deviation over 10 experiments. The upper curve gives RCC quality for combining results of all other 10 algorithms. The lower curve is the average quality of the other 10.

| data | subspace<br>#dims | #models<br>$r$ | **quality of consensus**<br>$\phi^{(\mathrm{NMI})}(\kappa, \lambda)$ | maximum quality in subspaces<br>$\max_q \phi^{(\mathrm{NMI})}(\kappa, \lambda^{(q)})$ | average quality in subspaces<br>$\mathrm{avg}_q \phi^{(\mathrm{NMI})}(\kappa, \lambda^{(q)})$ | minimum quality in subspaces<br>$\min_q \phi^{(\mathrm{NMI})}(\kappa, \lambda^{(q)})$ |
|---|---|---|---|---|---|---|
| `2D2K` | 1 | 3 | **0.68864** | 0.68864 | 0.64145 | 0.54706 |
| `8D5K` | 2 | 5 | **0.98913** | 0.76615 | 0.69822 | 0.62134 |
| `PENDIG` | 4 | 10 | **0.59009** | 0.53197 | 0.44625 | 0.32598 |
| `YAHOO` | 128 | 20 | **0.38167** | 0.21403 | 0.17075 | 0.14582 |

Table 2: FDC results. The consensus clustering is as good as or better than the best individual subspace clustering.

processing on the all features is not possible but multiple, limited views exist, a cluster ensemble can boost results significantly compared to individual clusterings.

## Concluding Remarks

In this paper we introduced the cluster ensemble problem and provide three effective and efficient algorithms to solve it. We conducted experiments to show how cluster ensembles can be used to introduce robustness and dramatically improve 'sets of subspace clusterings' for a large variety of domains. Selected data-sets and software are available from `http://strehl.com/`. Indeed, the cluster ensemble is a very general framework and enables a wide range of applications.

For future work, we desire a thorough theoretical analysis of the Average Normalized Mutual Information (ANMI) objective and its greedy optimization schemes. We also pursue work on comparing the proposed technique's biases. Moreover, we explore using cluster ensembles for object distributed clustering.

## References

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

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

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

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

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

Karypis, G.; Aggarwal, R.; Kumar, V.; and Shekhar, S. 1997. Multilevel hypergraph partitioning: Applications in VLSI domain. In *Proceedings of the Design and Automation Conference*.

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

Mehrotra, M. 1999. 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.

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

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

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

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

Strehl, A.; Ghosh, J.; and Mooney, R. 2000. Impact of similarity measures on web-page clustering. In *Proc. AAAI Workshop on AI for Web Search (AAAI 2000), Austin*, 58–64. AAAI/MIT Press.

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

Tumer, K., and Ghosh, J. 1996. Analysis of decision boundaries in linearly combined neural classifiers. *Pattern Recognition* 29(2):341–348.