A Consensus Framework for Integrating Distributed Clusterings Under Limited Knowledge Sharing

Joydeep Ghosh Alexander Strehl Srujana Merugu

Department of Electrical and Computer Engineering
The University of Texas at Austin
Austin, TX 78712, USA
E-mail: {ghosh,strehl,merugu}@ece.utexas.edu

Abstract

This paper examines 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. This problem is an abstraction of scenarios where different organizations have grouped some or all elements of a common underlying population, possibly using different features, algorithms or clustering criteria. Moreover, due to real life constraints such as proprietary techniques, legal restrictions, different data ownerships etc, it is not feasible to pool all the data into a central location and then apply clustering techniques: the only information that can be shared are the symbolic cluster labels. The cluster ensemble problem is formalized as a combinatorial optimization problem that obtains a consensus function in terms of shared mutual information among individual solutions. Three effective and efficient techniques for obtaining high-quality consensus functions are described and studied empirically for the following qualitatively different application scenarios: (i) where the original clusters were formed based on non-identical sets of features, (ii) where the original clustering algorithms were applied to non-identical sets of objects and (iii) when the individual solutions provide varying numbers of clusters. Promising results are obtained in all the three situations for synthetic as well as real data sets, even under severe restrictions on data and knowledge sharing.*

1. Introduction

In this paper, we address the problem of *combining* multiple *partitionings* of a set of objects *without accessing the original features*, in a distributed data mining context. This work starts from our very recently introduced framework for designing *cluster ensembles* [16, 14], and focusses on distributed computing scenarios where the combiner can only examine the cluster labels but not the original features or algorithms, and even the number of clusters obtained may differ from site to site. At first glance, this may be reminiscent of *classifier ensembles*, for which a large body of literature exists [13, 6, 11]. But actually the the cluster ensemble design problem is much more difficult 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 significantly from site to site due to differences in the clustering method used, the clustering objective, as well as on the particular view of the data available at those sites. Moreover, the desired number of clusters is often not known in advance, unlike in the typical classification setting.

1.1 Motivation.

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 datasets. 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 [15]. The company already had a variety of legacy customer segmentations based on demographics, credit rating, geographical region, purchasing patterns in 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 in the cluster ensemble problem, we provision for missing labels in individual clusterings.

Another application involving legacy solutions is segmenting of mortgage loan applicants based on the information in the application forms, supplemented by pre-existing groupings of the applicants indicated by proprietary external sources such as the FICO scores provided by Fairs Isaac.

Distributed Data Mining. 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 [9], 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. But this may not be desirable because of the computational, bandwidth and storage costs. In certain cases, it may not even be possible due to a variety of real-life constraints including security, privacy, proprietary nature of data and the accompanying ownership issues, need for fault tolerant distribution of data and services, real-time processing requirements, statutory constraints imposed by law, etc. [12]. Interestingly, the severity of such constraints is becoming very evident of late as several government agencies are attempting to integrate their databases and analytical techniques.

A cluster ensemble can perform *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, combination of DNA microarray data from many independent experiments, and mining of the biological literature (e.g., 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.

1.2 Related Work

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 a initial set of hierarchical clustering solutions [4]. 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 is presented in [3]. In all 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 [5]. The results were used to form a co-occurrence or similarity matrix that is is analogous to the one used for CSPA, except that the clusterings generated in [5] are not legacy but are generated from a common data set and feature space. Also they are at a much finer level of resolution than that desired by the final clustering, since the purpose of running multiple clusterings is to get a more robust similarity matrix that has the flavor of the shared nearest neighbor technique.

One use of cluster ensembles 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' [19], 'learning to learn' [20] and 'knowledge reuse' [2], but we have not seen these applied to totally unsupervised settings.

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 [9]. In [8] 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 [10], a distributed method of principal components analysis is introduced for clustering. The information sharing in these approaches is less restrictive than what is allowable in cluster ensembles.

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 (with possibly missing values) given a tuple of objects. Some clusterers may provide additional information such as description of cluster means, but we shall not use such information in this paper. Figure 1 shows the basic setup of the cluster ensem-

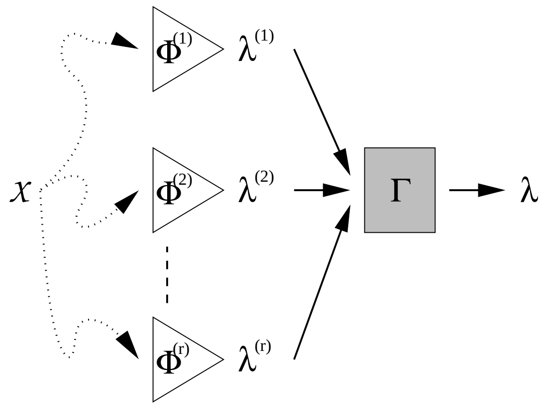


Figure 1. The Cluster Ensemble. A consensus function Γ combines clusterings $\lambda^{(q)}$ from a variety of sources, without resorting to the original object features $\mathcal{X}$ or algorithms Φ.

ble: A set of r labelings $\lambda^{(1,\ldots,r)}$ is combined into a single labeling λ (the *consensus labeling*) using a consensus function $\Gamma : \{\lambda^{(q)} \mid q \in \{1,\ldots,r\}\} \to \lambda$. A superscript in brackets denotes an index and not an exponent.

Organization. In the next section, we recapitulate our recent formulation of a cluster ensemble as an optimization problem [16]. A different normalization function is proposed that is more general in that it caters to both balanced and non-balanced clustering situations. Three effective and efficient combining functions Γ, as well as a direct optimization approach are then described and compared. In section 3, we describe applications of cluster ensembles for the distributed data mining scenarios described above, and show results on both real and artificial data. Section 4 concentrates on knowledge reuse when legacy clusters are of variable resolution.

2. Cluster Ensembles

Objective Function. Let the set of groupings $\{\lambda^{(q)} \mid q \in \{1,\ldots,r\}\}$ be denoted by $\mathbf{\Lambda}$, and let the q-th grouping have $k^{(q)}$ clusters. If there is no apriori 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, provides a sound indication of the shared information between a pair of clusterings. Let $I(X,Y)$ denote the mutual information between two random variables X and Y, and $H(X)$ denote the entropy of X. Since 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. In [16] we used a normalization suitable when the consensus clustering needs to be balanced, i.e., the clusters should be of comparable sizes. We now propose a more general approach based on the existence of a Hilbert Space with vectors $\mathbf{x}$ and $\mathbf{y}$ representing the random variables X and Y, such that the inner product measure $<\mathbf{x},\mathbf{y}>$ is identical to $I(X,Y)$. Then the natural normalization to the range [0,1] is $\frac{<\mathbf{x},\mathbf{y}>}{\sqrt{<\mathbf{x},\mathbf{x}>,<\mathbf{y},\mathbf{y}>}}$. Noting that $I(X,X) = H(X)$, we get a normalized mutual information (NMI):

$$NMI(X,Y) = \frac{I(X,Y)}{\sqrt{H(X)H(Y)}} \tag{1}$$

For consensus clustering, the random variables are represented by the cluster labelings $\lambda^{(a)}$ and $\lambda^{(b)}$, with $k^{(a)}$ and $k^{(b)}$ groups respectively, and Equation 1 needs to be estimated by the sampled quantities provided by the clusterings. We denote the resulting normalized mutual information measure by $\phi^{(\mathrm{NMI})}(\lambda^{(a)},\lambda^{(b)})$.

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):

$$\phi^{(\mathrm{ANMI})}(\mathbf{\Lambda},\hat{\lambda}) = \frac{1}{r}\sum_{q=1}^{r}\phi^{(\mathrm{NMI})}(\hat{\lambda},\lambda^{(q)}). \tag{2}$$

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, i.e.,

$$\lambda^{(k-\mathrm{opt})} = \arg\max_{\hat{\lambda}} \phi^{(\mathrm{ANMI})}(\mathbf{\Lambda},\hat{\lambda}), \tag{3}$$

where $\hat{\lambda}$ goes through all possible k-partitions. 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 are missing data in the label vectors. For such cases, the consensus clustering objective from equation 3 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.

2.1. Consensus Function Heuristics

In [16, 14], we introduced 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. Essentially each object is a vertex, and all the members of a given cluster in any solution are connected by an hyperedge. So the total number of hyperedges is simply the sum of the number of clusters over all r clusterings. Based on this representation, the following heuristics were proposed (details in [16]):

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.

2.2. Direct Optimization Approaches

Instead of designing heuristics, one can try to directly optimize the objective functions in Eq. 3. First, note that 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 $\frac{1}{k!}\sum_{\ell=1}^{k}\left(\begin{array}{c}k\\ \ell\end{array}\right)(-1)^{k-\ell}\ell^{n}$ possible clusterings, or approximately $k^n/k!$ for $n \gg k$ [7]. For example, there are 171,798,901 ways to form 4 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. Typically such approaches are too computationally expensive to be relevant in a data mining context. 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 hours per run on a 1 GHz PC using our implementation.

2.3. Discussion and Comparison

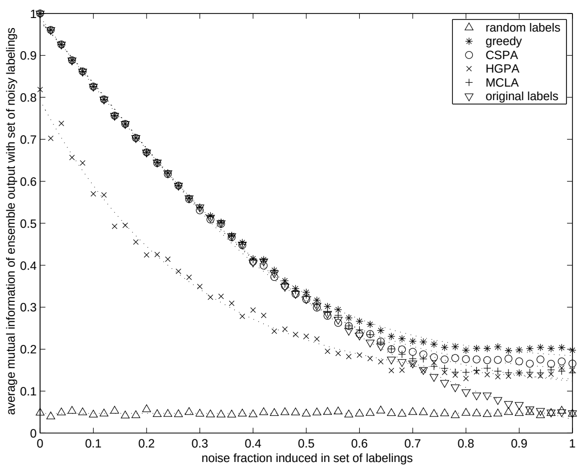
In this section, we compare the heuristics with the direct approach, and also do a sanity check to see if the NMI criteria is indeed meaningful. Let us first take a look at the worst case time complexity of the proposed algorithms. Assuming quasi-linear (hyper-)graph partitioners such as (H)METIS, CSPA 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 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.[1] 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. Firstly, we measure the normalized objective function $\phi^{(\mathrm{ANMI})}(\mathbf{\Lambda}, \lambda)$ of the ensemble output λ with all the individual labels in $\mathbf{\Lambda}$. Secondly, 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.[2]

Figure 2 shows the results. As noise increases, labelings share less information and thus maximum obtainable $\phi^{(\mathrm{ANMI})}(\mathbf{\Lambda}, \lambda)$ decreases, and so does $\phi^{(\mathrm{ANMI})}(\mathbf{\Lambda}, \lambda)$ for all techniques (figure 2(top)). HGPA performs the worst in

[1]Labels are obtained by a random permutation. Groups are balanced.

[2]In low noise settings, the original labels are the global maximum, since they share the most mutual information with the distorted labelings.



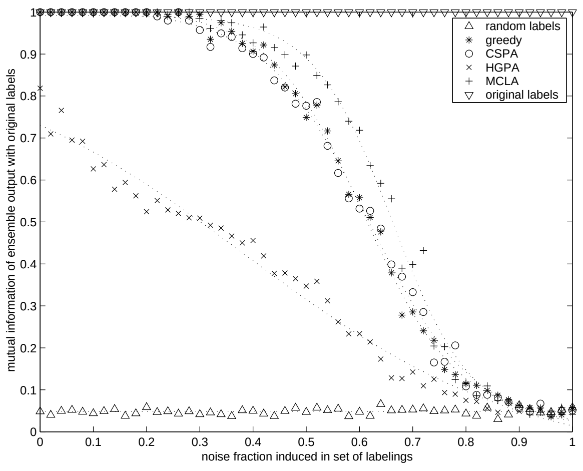


Figure 2. Comparison of consensus functions in terms of $\phi^{(\mathrm{ANMI})}(\mathbf{\Lambda}, \lambda)$ (top) and in terms of $\phi^{(\mathrm{NMI})}(\kappa, \lambda)$ (bottom) for various noise levels. A fitted sigmoid (least squared error) is shown for all algorithms to show the trend.

this experiment, which we believe is due to the lacking provision of 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 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 are almost completely random and the consensus algorithms recover whatever little common information is present whereas the original labeling is now almost fully unrelated. However, realistically noise should not exceed 50% and MCLA seems to perform best in this simple controlled experiment.

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. Since in a real setting noise is expected to be much less than 50%, 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 less 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.

A Supra-Consensus Function. Our objective function has an added advantage that it allows one to add a stage that selects the best consensus function without any supervision 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, and selecting the one with the greatest ANMI. Then, if there are significant differences or notable trends 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 dataset beforehand.

3. Empirical Studies

Data Sets. We illustrate the cluster ensemble applications on two real and two artificial data-sets. In table 1 some basic properties of the datasets (left) and parameter choices (right) are summarized. (2D2K) is the simplest, containing 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 artificial data-set, 8D5K, contains

1000 points from 5 multivariate Gaussian distributions (200 points each) in 8D space. Again, clusters all have the same variance (0.1), but different means. Both artificial data-sets are available for download at `http://strehl.com/`.

The third data-set (`PENDIG`) for pen-based recognition of handwritten digits is taken from the UCI Machine Learning Repository. 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, `YAHOO`, is for text clustering and contains 20 original Yahoo! news categories. The data is publicly available from `ftp://ftp.cs.umn.edu/dept/users/boley/` (K1 series) and was used in [1, 17]. The raw 21839×2340 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 (e.g., `haruspex`) or too generic (e.g., `new`), respectively, results in $d = 2903$. The default k used for `YAHOO` is taken as 40 (two times the number of categories), since some categories seem to be multi-modal.

Evaluation Criterion. Evaluation of the quality of a clustering is a non-trivial and often ill-posed task. In fact, many definitions of objective functions for clusterings exist [7]. Since for our data sets either the generative models or the class labels are known, we can use an extrinsic measure based on comparing category labels to class labels, as opposed to intrinsic measures such as compactness and separation. Normalized mutual information is chosen as it is impartial with respect to k as compared to other extrinsic criteria such as purity and entropy. It reaches its maximum value of 1 only when the two sets of labels have an exact one-to-one correspondence.

3.1. Feature-Distributed Clustering (FDC)

In Feature-Distributed Clustering (FDC), we show how cluster ensembles can be used to combine a set of clusterings obtained in a distributed environment from partial views of the data. We run several clusterers, each having access to only a restricted, small subset of features. Note that because of the current lack of public domain datasets for distributed clustering, in our experiments these partial views had to be created from a common feature space. In a real-life scenario, the different views would be determined *apriori* 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.

Table 2 summarizes the results. Each dataset was projected onto much lower dimensional, randomly chosen subspaces, for r times, a graph-partition based clustering was done on each projection, and the results integrated using the supra-consensus function. 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 0.16 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% 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. Also, the supra-consensus function chooses either MCLA and CSPA results, but the difference is not statistically significant.

3.2. Object-Distributed Clustering (ODC)

A dual to 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 the features of the objects they are provided with. 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[3].

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.

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. For experimental purposes, we divide the data into p overlapping partitions such that on an average, each object resides in v partitions. For simplicity, each partition is of the same size, i.e. nv/p. 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. 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.

Figure 3 shows our results for the four data-sets when graph partitioning was used as the clusterer in each processor. Each plot in figure 3 shows the relative mutual informa-

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

name	features	#features	#categories	balance	similarity	default #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 1. Overview of datasets for cluster ensemble experiments. Balance is defined as the ratio of the average category size to the largest category size.

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 $\mathrm{avg}_q \phi^{(\mathrm{NMI})}(\kappa, \lambda^{(q)})$	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 2. FDC results. The consensus clustering is as good as or better than the best individual subspace clustering.

tion (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 $\circ$. 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.

Distributed clustering using a cluster ensemble also provides a speedup when the inner loop clustering algorithm has superlinear complexity ($> O(n)$) and a fast consensus function (such as MCLA and HGPA) is used. For example, let us assume that the inner loop clusterer has a complexity of $O(n^2)$ (e.g., similarity-based approaches or efficient agglomerative clustering) and one uses only MCLA and HGPA in the supra-consensus function.[4] 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 e.g., 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, 2D2K (YAHOO) can be sped up 64-fold using 16 processors at 90% (80%) of the full length quality.

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

4. Integrating Clusterings of Varying Resolutions

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? To address this question, we ran experiments where clusterings over a range of k were integrated. While one would expect that different sites are looking at different features if they select different values of k, to remove this additional source of variability, we just used the same set of features for all clusterings. Also, since 2D2K is very simple, we replaced it with NEWS20, which consists of newsgroup data with 20 categories, available from `http://www.ai.mit.edu/people/jrennie/20Newsgroups/`. We sampled the original dataset to get a reduced data set of 2000 documents (100/category), each represented by a 1151 dimensional feature vector.

Table 3 summarizes the experiments performed. Spherical K-means is used for the last two datasets because they are so high-dimensional and non-Gaussian that regular K-means performs miserably on them [18]. Ensemble-A indicates the original ranges of k chosen. We found that, given a wide range of k, ANMI would typically peak around the most appropriate value of k, hence it could be used to narrow down the range of k for re-consideration by the consensus function. This observation results in Ensemble-B,

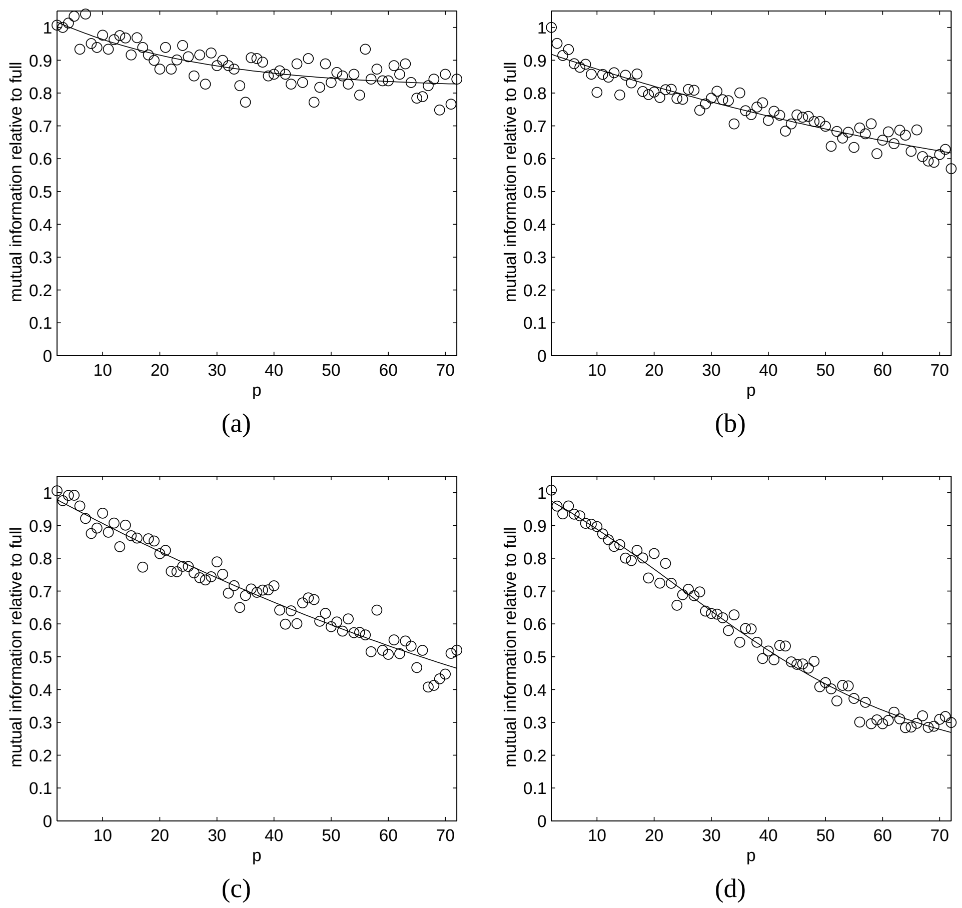


Figure 3. ODC Results. Clustering quality (measured by relative mutual information) as a function of the number of partitions, p, on various data sets: (a) `2D2K`; (b) `8D5K`; (c) `PENDIG`; (d) `YAHOO`. The sum of the number of samples over all partitions is fixed at $2n$. Each plot contains experimental results using graph partitioning in the inner loop for $p = [2, \ldots, 72]$.

where we zoom into a narrower range of k. For example, for `8D5K`, for the first set of k values, the average mutual information varies quite smoothly and peaks around $k = 6$, as shown in the Figure 4. So we chose the new range of k to be [4-8] and we find that $k = 5$ gives the highest ANMI value. Incidentally, this highlights an added benefit of using a cluster ensemble, since they give a good indication of the natural number of clusters in the data.

First, as a sanity check, we plotted the values of average mutual information with all the members of a cluster ensemble and the mutual information with the original clustering (Figure 5). The correlation coefficient, averaged over all the eight cluster ensembles in the experiments is 0.9023, which is quite high. This justifies our approach of picking solutions based on their average mutual information with respect to the cluster ensemble.

Table 4 shows the mutual information values between the different clusterings and the original categorization of the corresponding data set. For each data set, the “A” and “B” versions indicate which ensemble was chosen, and only the supra-consensus results are shown for brevity. The average clustering quality is obtained by computing the average of the pairwise mutual information of each ensemble member with the original categorization. The natural-k clusterings are the solutions obtained by directly applying the clustering algorithm to get natural-k clusters. The max-ANMI clusterings, on the other hand refer to the cluster ensemble members that share the maximum mutual information with all the members of the ensemble.

From the table, we notice that the quality of the consensus clustering is better than that of the natural-k clustering, the max-ANMI clustering and also the average quality of the ensemble itself, except in the case of the YAHOO dataset. This indicates that the consensus method is a good way to obtain a high quality clustering when the number of clusters is not known. Furthermore, the quality of a max-ANMI clustering is also higher than the average of the corresponding ensemble. So the max-ANMI clustering can be used to obtain a moderately good solution when it is not possible or expensive to do consensus.

The deviation in the case of the YAHOO dataset is most likely because the original manually assigned categories are somewhat different from the natural clustering of the data, and the categories are highly imbalanced as well. The low NMI values of the YAHOO clusterings as well as the lower correlation value (0.6086) between the Average NMI and the NMI values in the YAHOO cluster ensembles seem to substantiate this. Hence, it is not very surprising that the clustering obtained using the consensus methods doesn't perform very well for this dataset.

Concluding Remarks. Cluster ensembles enable federated data mining systems working on top of distributed and heterogeneous databases, even under severe data and knowledge sharing constraints. We are actively looking for more real-life data sets to extend our application scenarios and do further comparative studies.

Acknowledgements. We would like to thank Intel Corp and IBM ACAS for their generous support of this work.

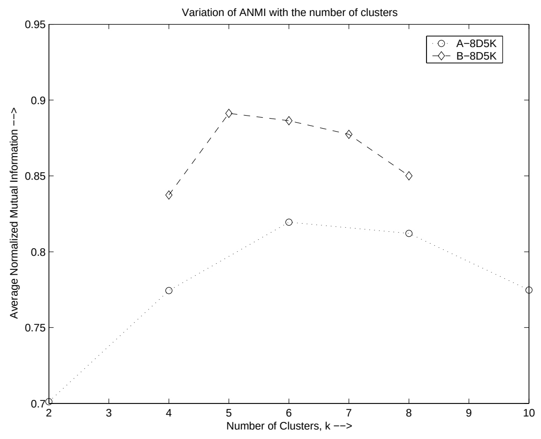


Figure 4. Plot showing the variation of ANMI with respect to the number of clusters k.

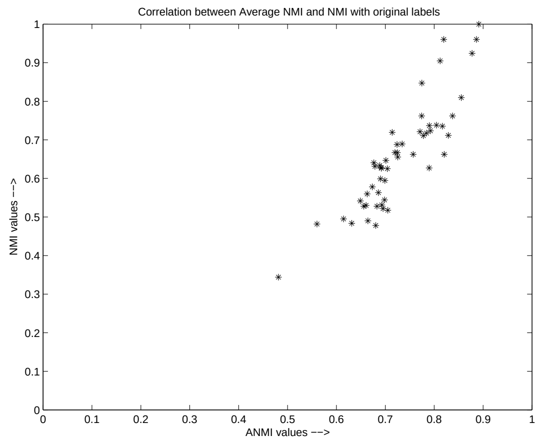


Figure 5. Plot of the cluster quality values with respect to the average mutual information shared with the members of the ensemble

References

[1] 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.

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

[3] 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.

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

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

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

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

[8] 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.

DataSet	Clust. Algo.	Similarity	Natural-k	Ensemble-A	Ensemble-B
8D5K	K-Means	Euclidean	5	$k = [2:2:10]$	$k = [4:1:8]$
PENDIG	K-Means	Euclidean	10	$k = [2:4:30]$	$k = [8:1:12]$
NEWS20	Spherical K-Means	Cosine	20	$k = \{5\} \bigcup [10:10:60]$	$k = [14:2:26]$
YAHOO	Spherical K-Means	Cosine	20	$k = \{5\} \bigcup [10:10:60]$	$k = [14:2:26]$

Table 3. Details of the datasets and cluster ensembles with varying k. The notation $k = \{a : b : c\}$ indicates that k ranges from a to c in steps of b.

Ensemble	Consensus	Average	Natural-k	Max-ANMI
A-8D5K	1.0000	0.8242	1.0000	0.9603
B-8D5K	1.0000	0.8921	1.0000	1.0000
A-PENDIG	0.7023	0.6514	0.6624	0.7035
B-PENDIG	0.6800	0.6641	0.6624	0.6624
A-NEWS20	0.6723	0.6282	0.6438	0.6689
B-NEWS20	0.7029	0.6559	0.6438	0.6881
A-YAHOO	0.4998	0.5267	0.5097	0.5437
B-YAHOO	0.5017	0.5224	0.5097	0.5442

Table 4. Normalised Mutual Information of the clusterings with respect to the corresponding original categorization.

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

[10] 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.

[11] J. Kittler and F. Roli (Eds.). *Multiple Classifier Systems*. LNCS Vol. 1857, Springer, 2001.

[12] 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.

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

[14] A. Strehl and J. Ghosh. Cluster ensembles – a knowledge reuse framework for combining partitionings. *JMLR, under review*.

[15] A. Strehl and J. 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.

[16] 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 2002.

[17] A. Strehl, J. Ghosh, and R. 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.

[18] A. Strehl, J. Ghosh, and R. Mooney. Impact of similarity measures on web-page clustering. In *Proceedings of Seventeenth National Conference on Artificial Intelligence : Workshop of Artificial Intelligence for Web Search (AAAI 2000), 30-31 July 2000, Austin, Texas, USA*, pages 58–64. AAAI, July 2000.

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

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