Pattern Recognition Letters 29 (2008) 1947–1953

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

# Ensemble clustering with voting active clusters

Kagan Tumer [a,*], Adrian K. Agogino [b]

[a] Oregon State University, 204 Rogers Hall, Corvallis, OR 97330, USA
[b] UCSC, NASA Ames Res. Ctr., Mailstop 269-3, Moffett Field, CA 94035, USA

ARTICLE INFO

*Article history:*
Received 4 October 2007
Received in revised form 16 April 2008
Available online 25 June 2008

Communicated by F. Roli

*Keywords:*
Cluster ensembles
Consensus clustering
Distributed clustering
Adaptive clustering

ABSTRACT

Clustering is an integral part of pattern recognition problems and is connected to both the data reduction and the data understanding steps. Combining multiple clusterings into an ensemble clustering is critical in many real world applications, particularly for domains with large data sets, high-dimensional feature sets and proprietary data. This paper presents voting active clusters (VACs), a method for combining multiple "base" clusterings into a single unified "ensemble" clustering that is robust against missing data and does not require all the data to be collected in one central location. In this approach, separate processing centers produce many base clusterings based on some portion of the data. The clusterings of such separate processing centers are then pooled to produce a unified ensemble clustering through a voting mechanism. The major contribution of this work is in providing an adaptive voting method by which the clusterings (e.g., spatially distributed processing centers) update their votes in order to maximize an overall quality measure. Our results show that this method achieves comparable or better performance than traditional cluster ensemble methods in noise-free conditions, and remains effective in noisy scenarios where many traditional methods are inapplicable.



## 1. Introduction

Forming clusters in a data set that provides a meaningful grouping is a difficult problem and is addressed in numerous fields including pattern recognition, machine learning, data management and bioinformatics (Kim and Warnow, 1999; Mitchell, 1997; Witten and Frank, 2005). The central concept of clustering is that data points should be partitioned into separate clusters so that data points within a cluster are "close" to each other, while data points from different clusters are "far" from each other. How close two points are to one another is based on distance metrics that capture the important aspects of the domain (e.g., document clustering might look at words in common, and market basket clustering for online shoppers might look at similarity of purchases).

A partition of the data into a set of clusters is a clustering, and the clustering problem consists of finding a "good" clustering. For an $N$ point data set $\{X_1,\ldots,X_N\}$ where each point consists of $F$ features ($X_j = \{X_{j,1},\ldots,X_{j,F}\}$), a particular clustering $Y_k$ results in labels $\{Y_{1,k},\ldots,Y_{N,k}\}$ being associated with each data point. These labels assign each original data point into a particular cluster. Different clustering computations (e.g., different processing centers $C_k$ having potentially access to different components of $\{X_1,\ldots,X_N\}$) typically produce different clusterings $Y_k$ (see Fig. 1). While the clustering problem is an NP-complete optimization problem, numerous heuristic algorithms (e.g., k-means, expectation-maximization and METIS) provide satisfactory solutions in many applications (Bishop, 1996; Chakravarthy and Ghosh, 1996; Karypis and Kumar, 1998; Kernighan and Lin, 1970; MacQueen et al., 1967).

Such algorithms are based on the assumption that all of the data, $\{X_1,\ldots,X_N\}$, is available at a central location at the same time. In many real world distributed computational problems, such as processing sensor suites coming from satellite arrays, data are often collected, pre-processed and clustered on separate nodes of computations ($C_1,\ldots,C_S$). Still, the ultimate objective is to provide a combined clustering to form a unified view of the entire system. A straight forward approach to this problem requires one to aggregate all the data from all of the sources and perform a unified clustering computation on the full data set. However, clustering a full set of data in this way is often undesirable for a number of reasons.

First, collecting all the data (e.g., all of the features for all the patterns) in one place may be impractical or impossible. For instance in document clustering a data point is an entire document. While it may not be feasible to put the entire document corpus ($\{X_1,\ldots,X_N\}$) all in one place to perform clustering, it may be possible to have separate clusterings of a subset of the documents ($\{Y_{1,1},\ldots,Y_{N,S}\}$) and then combine the clusterings. Second, some or part of that data may be proprietary. In such a case, revealing clusters may be an option whereas revealing the data are not. For example a credit card company may be willing to share summary information with some agency, but not divulge the full data

* Corresponding author.
*E-mail addresses:* kagan.tumer@oregonstate.edu (K. Tumer), adrian@email.arc.nasa.gov (A.K. Agogino).

0167-8655/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.patrec.2008.06.011

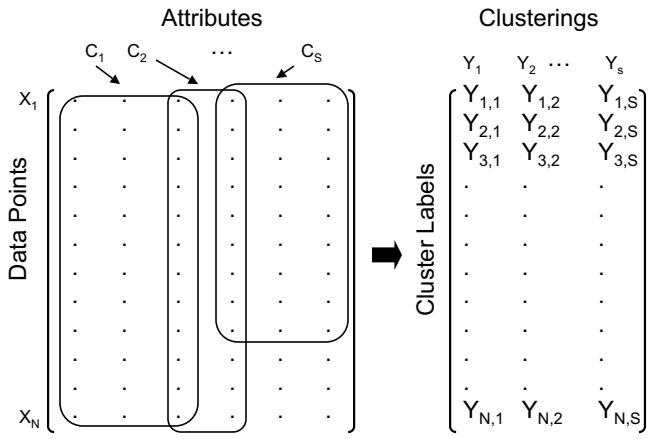


**Fig. 1.** *Cluster ensemble problem.* $N$ data points $\{X_1,\ldots,X_N\}$, are clustered by $S$ base-clusterings (data processing centers) $\{C_1,\ldots,C_S\}$, resulting in $NxS$ labels $\{Y_{1,1},\ldots,Y_{N,S}\}$. The cluster ensemble problem consists of providing the best "ensemble clustering", $Z$, for a given metric, using only the $Y_{i,j}$ labels provided by the base clusterings $\{Y_i\}$.

containing all the purchases of each of its clients. Finally, the data may be collected sequentially, resulting in "temporal separation". Waiting until all the data are present fully at one location at one time may lead to unnecessary delays. Using only cluster labels from earlier time slices to speedily summarize the data may be advantageous in such a case.

To address such issues, in this article, instead of requiring data aggregation to perform clustering operations, we focus on the "cluster ensemble" problem where pre-existing "base" clusterings ($\{Y_{1,1},\ldots,Y_{N,S}\}$) computed on the separate data sources ($C_1,\ldots,C_S$) are combined into a single unified "ensemble" clustering, $Z$ (Agogino et al., 2006; Minaei-Bidgoli et al., 2004; Strehl et al., 2002b; Topchy et al., 2003).[1] In this solution, voting active clusters (VACs) vote on how their data points should be labeled in the final clustering and receive feedback based on the quality of the final ensemble clustering. Through this feedback mechanism VACs learn to vote for labels that lead to a good ensemble clustering. The advantage of the VACs solution is that it is naturally distributed and elegantly matches the needs of a distributed real world system in that it is robust against noisy data sources; naturally adapts to loss of data sources; and performance gracefully degrades with the number of failures. These advantages allow cluster ensembles to be used more effectively in more domains than their graph-partitioning based counterparts, and open the possibility of using cluster ensembles in many real world applications.

In Section 2, we provide a brief overview of graph-based and simple greedy optimization-based solutions to the cluster ensemble problem. In Section 3, we describe the cluster ensemble problem formally. In Section 4 we introduce the VACs approach. In Section 5 we presents results in an artificial domain comparing the new approach to traditional approaches. In Section 6 we apply the VACs approach to two real world problems. Finally, in Section 7 we summarize our findings and discuss the implication of our results.

## 2. Related work

There are two broad classes of traditional approaches to the cluster ensemble problem: graph-based methods and greedy optimization methods. In this section we provide a brief summary of both sets of approaches.

### 2.1. Graph-based methods

The graph-based approaches to the cluster ensemble problem involve representing the data points as edges in a hypergraph, and the clusters as undirected hyperedges of the hypergraph.[2]

Three of the most widely used graph-based cluster ensemble algorithms are:

- *Cluster-based Similarity Partitioning Algorithm (CSPA)*: This algorithm creates a similarity measure between data points. CSPA provides moderate performance, but requires significant computation (Strehl et al., 2002b).
- *HyperGraph Partitioning Algorithm (HPGA)*: This algorithm creates the final ensemble clustering by using a hypergraph partitioning algorithm (HMETIS (Karypis et al., 1997)) to partition the hypergraph that represents the clusterings. It is simpler to compute than CSPA (Strehl et al., 2002a). The objective of this method is to create clusters that break the least number of hyperedges.
- *Meta-CLustering Algorithm (MCLA)*: This algorithm operates by collapsing a group of related hyperedges into a single hyperedge. This provides better performance than HPGA and retains its low computational complexity (Strehl et al., 2002b).

Graph-based methods require centralized computation, where all of the cluster labels are centralized and processed in one location (Strehl et al., 2002b). While these methods work well in ideal conditions, they are not well suited for certain, large real world domains. First, graph based methods are not robust and provide poor results if any of the clusterings provide corrupted cluster labels. Second, in many real world problems, data sources are often unreliable, going on and offline frequently. In addition due to communication problems, parts of the data are often noisy or absent. In such cases, traditional algorithms either require cumbersome redundancy mechanisms or fail catastrophically. A method that can maintain adequate levels of performance when parts of the system is cut off from the rest can also increase the appeal of clustering algorithms.

### 2.2. Greedy optimization

Simple greedy optimization approaches have been applied to the cluster ensemble problem in (Strehl et al., 2002b). In such methods, one starts with a single representative clustering, $Z$, which is the clustering most similar to all the other clusterings. In general, this means, $Z$ maximizes a function:

$$Z = \text{argmax}_{Y_i \in \overline{Y}} \Phi(\overline{Y} \setminus Y_i, Y_i), \quad (1)$$

where $\overline{Y}$ is the set of clusterings, and $\setminus$ is the set difference operator, and $\Phi$ is a function measuring the similarity between its arguments (for example mutual information, see Section 3 for details). This clustering $Z$ is chosen out of the existing clusterings and thus is only a crude approximation to $Z^*$, the clustering which has the highest possible value for $\Phi(\overline{Y}, Z)$.

To try to find a clustering with higher $\Phi$, a new clustering is created from $Z$ by randomly moving a data point to a new cluster. The resulting clustering is only accepted if it has a higher $\Phi$ value, and the process is repeated for each data point. The algorithm stops when it cycles through all the data without accepting a new clus-

[1] Note that besides cluster ensembles, generative models can also be used to combine clusterings, which tradeoff performance with the amount of information and therefore data privacy that is needed (Merugu et al., 2003). However this paper will focus on the ensemble method where only the cluster labels are used.

[2] A hyperedge is an edge that connects more than two nodes. A hypergraph is a graph with hyperedges (Karypis et al., 1997).

tering (e.g., all data moves result in a worse clustering than the current one). This algorithm is a basic local search algorithm, with no exploration (i.e., it can be seen as a form of serial simulated annealing with zero exploration). In addition to suffering from local minima due to the lack of exploration, it has a high computational burden: For each loop, $\Phi$ (usually a function expensive to compute) needs to be computed for each data point. This makes the algorithm computationally burdensome in large real world data sets.

## 3. Cluster ensemble problem

The clustering problem addressed in this article is equivalent to creating a non-overlapping partition of the data, and a particular partition is referred to as a *clustering*. Because in this case a data point can only belong to a single cluster, a clustering can be seen as a labeling of the data, where each data point is assigned a label showing the cluster to which it belongs. In general, there can be many different clusterings for the same data, depending on the clustering algorithm used (Dhillon and Modha, 2001; Karypis and Kumar, 1998; Kohonen, 1982). The different clusterings analyze different aspects of the data and result in different biases in the clustering algorithms. The goal of the cluster ensemble problem is to create a single clustering that best characterizes a set of clusterings, without using the original data points used to generate the base clusterings. Cluster ensembles allow the user a lot of flexibility in analyzing the data and as with classifier ensembles often lead to better and more robust solutions.

To formalize the clustering ensemble problem, we need a way to compare two clusterings. Though different metrics exist (Fowlkes and Mallows, 1983; Hubert and Arabie, 1985), in this paper, we focus on an information theoretic measure previously used in the clustering community called *normalized mutual information* (NMI) (Fern and Brodley, 2004; Strehl et al., 2002b). This measure is based on the sizes of the clusters within the clusterings (a formal justification for this is presented in (Strehl et al., 2002a). To compute this measure, let us define $X_c^i$ as the set of points in $\{X\}$ that are in the same cluster $c$ in clustering $Y_i$ (that is $X_c^i : \{X_l \text{s.t.} Y_{l,i} = c\}$). We now define the mutual information between clusterings $Y_i$ and $Y_k$ as

$$I(Y_i, Y_k) = \sum_{X_c^i \in Y_i} \sum_{X_{c'}^k \in Y_k} \frac{|X_c^i \cap X_{c'}^k|}{N} \log_2 \left( \frac{N |X_c^i \cap X_{c'}^k|}{|X_c^i||X_{c'}^k|} \right), \tag{2}$$

where there are $N$ total data points and $|\cdot|$ and $\cap$ are the set cardinality and intersection operators, respectively. The intersection operation $X_c^i \cap X_{c'}^k$ returns the data points that are common to both data point clusters $X_c^i$ and $X_{c'}^k$. Next we define the entropy of a clustering, $Y_i$, as

$$H(Y_i) = -\sum_{X_c^i \in Y_i} \frac{|X_c^i|}{N} \log_2 \left( \frac{|X_c^i|}{N} \right). \tag{3}$$

Now we can define the normalized mutual information (NMI) between two clusterings, $Y_i$ and $Y_k$ as

$$\text{NMI}(Y_i, Y_k) = \frac{I(Y_i, Y_k)}{\sqrt{H(Y_i) H(Y_k)}}, \tag{4}$$

which has the desirable properties of having $\text{NMI}(Y, Y) = 1$ and being bounded by $[0, 1]$.

The normalized mutual information between two clusters measures how much information they have in common. If two clusterings are similar they will have an NMI close to one. In this case if we knew one clustering we would be able to predict most of the cluster assignments in the other clustering. For example if the same data set was clustered twice with the same clustering algorithm with only minor changes to parameters, the NMI of the two clusterings would likely be close to one. In contrast, if two clusterings are completely different then they have an NMI of zero. In this case knowing one clustering would not help predict the clusters within the second clustering. For example if two clusterings were produced with two completely different interpretations of the data, we would expect the NMI between the two clusterings to be close to zero.

If we had a "true" clustering of the data, we would want our ensemble clustering to have as high as possible NMI with respect to this true clustering. However, typically all we have to compare our ensemble clustering with is the base clusterings used to create the ensemble. In this case our goal is to create an ensemble clustering that shares as much information as possible with the base clusterings from which it was created. This can be measured by averaging the normalized mutual information between the base clusterings and the ensemble. Formally we define the ANMI between a clustering $Y_i$ and a set of clusterings $\overline{Y}$ as (Strehl et al., 2002a)

$$\text{ANMI}(\overline{Y}, Y_i) = \frac{1}{|\overline{Y}|} \sum_{Y_k \in \overline{Y}} \text{NMI}(Y_i, Y_k). \tag{5}$$

When an ensemble clustering $Z$ has more information in common with the base clusterings $\overline{Y}$, $\text{ANMI}(\overline{Y}, Z)$ will have a higher value. Therefore the objective is to find the clustering, $Z^*$, that maximizes the ANMI between $Z^*$ and the set of available clusterings $\overline{Y}$.

## 4. Voting active clusters for ensemble clustering

The VACs approach treats the cluster ensemble problem as a dynamic optimization problem, with the system striving to create a final combined clustering that maximizes the ANMI. In this approach, each cluster in the original set of base clusterings gets one vote per data point. Then the learning system for $r$ clusterings with $k_i$ clusters each has $\sum_{i=1}^{i=r} k_i$ VACs. The "action" of a VAC is to vote on the cluster in the ensemble clustering, to which its data points should belong. During a round of voting, a VAC only casts a single vote and all of the data points within that VAC are labeled with that vote. A data point will then belong to the final cluster that received the most votes from VACs that contained the data point.

### 4.1. Learning in voting active clusters

The votes of the VACs are adapted using a simple reinforcement learning algorithm (Kaelbing et al., 1996; Sutton and Barto, 1998). Reinforcement learning is a broadly used learning method and recent advances have made it especially amenable to distributed computation (Dietterich, 2000; Sutton and Barto, 1998; Tumer and Wolpert, 2004; Zhang and Dietterich, 1995). In general, a reinforcement learner takes actions and receives rewards evaluating the value, or quality, of the actions. The goal of a reinforcement learner is to learn to take actions that lead to high values of its reward. Reinforcement learning is an iterative process where at first the learners take almost random actions, but are continually exploring their action spaces to find actions that lead to high values of the reward function (Kaelbing et al., 1996; Sutton and Barto, 1998). In this work, we use the following algorithm:

1. Initialize action policy to random
2. Take action $a$ based on current action value
   - with probability $\epsilon$ choose random $a$
   - otherwise choose $a$ with highest expected value
3. Receive reward $R$ based on action taken
4. Modify action values based on Eq. 6
5. Go to step 2

The votes of the VACs ultimately determine the final clustering and the rewards are based on an evaluation of the final clustering. While many reinforcement learning systems attempt to learn a sequence of actions that lead to a high aggregate value of the objective function over time, for the purposes of our algorithm we only need simple reinforcement learners that try to take actions that maximize their immediate reward. In this case, a policy is simply a table having the same size as the action space. After a vote $a$ is taken and a reward $R$ is received, the value $V_a$, which is associated with the expected reward resulting from vote $a$ is modified by a VAC as (Sutton and Barto, 1998)

$$V_a^{\text{new}} = \alpha R + (1-\alpha)V_a^{\text{old}}, \tag{6}$$

where $\alpha$ is the learning rate. In this manner, under benign assumptions, the value of taking action $a$, $V_a$ (i.e., the expected reward), converges to the true reward of choosing vote $a$ (Kaelbing et al., 1996; Sutton and Barto, 1998).

### 4.2. Rewards for voting active clusters

The most direct choice for the reward the active clusters receive is the ANMI between the base clustering and, $Z(V)$, the clustering resulting from the votes $V$ of the voting active clusters (we will denote this reward received by all voting active clusters by $\text{ANMI}(\overline{Y}, Z(V))$. Since typically the "true clusterings" are not known this is a good global performance measure rating the performance of the full system. Note that in our test problems below we do know the "true clustering"; therefore though the algorithms will use the ANMI to train, the methods will be evaluated based on the "global performance" using the NMI between $Z(V)$ and the true clustering.

While VACs can use this ANMI directly to learn the best votes, it is difficult for a VAC to maximize a function that depends on the actions of that many variables. When there are many VACs (e.g., many clusters within many clusterings), it will take a large number of learning steps for a VAC to discern the effects of its vote from the votes of all the other VACs. Instead of maximizing the ANMI directly, a VAC can learn more quickly if its reward function is more sensitive to its own votes than the votes of all the other VACs.

To overcome this difficulty, in addition to adapting the votes of a VAC using the ANMI, we present an alternative VAC reward function, the Difference Normalized Mutual Information (DNMI):[3]

$$\text{DNMI}_i(\overline{Y}, Z(V)) = \text{ANMI}(\overline{Y}, Z(V)) - \text{ANMI}(\overline{Y}, Z(V')) \tag{7}$$

where $V'$ is a "counterfactual" vote total computed by removing VAC $i$'s vote from the system.

Because the second term in the right hand side of this equation does not depend on the vote of VAC $i$, a VAC maximizing $\text{DNMI}_i(\overline{Y}, Z(V))$ will tend to maximize $\text{ANMI}(\overline{Y}, Z(V))$ (e.g., they have the same derivative). However, a VAC will learn more quickly using $\text{DNMI}_i(\overline{Y}, Z(V))$ since each VAC has more influence over its own DNMI than over ANMI. Note that when a VAC's vote does not influence the final clustering $Z(V')$ the corresponding DNMI does not have to be computed since its value is zero. In large systems, this case will occur often. In addition a VAC also only needs the vote information on the data points it contains to determine if it made a deciding vote, significantly reducing the communication burden. If a VAC's vote is a deciding vote, then all the VACs will have to compute and broadcasts their NMIs for the computation of the clustering $Z(V')$ that would result from the counterfactual vote $V'$.

[3] This "difference" reward function has proven effective in many distributed learning domains (e.g., multi-agent systems) including network routing, rover control, job scheduling and congestion games (Agogino et al., 2004; Tumer and Wolpert, 2004; Tumer et al., 2000; Wolpert and Tumer, 2001).

**Table 1**
Global performance (NMI) on artificial data ($B = 0$)

| Algorithm | Performance | Standard deviation |
|---|---|---|
| ***MCLA*** | ***1.0*** | – |
| ***CSPA*** | ***1.0*** | – |
| HPGA | 0.81 | – |
| ***G. OPT*** | ***1.0*** | – |
| ***VAC (DNMI)*** | ***1.0*** | ***±0.0*** |
| VAC (ANMI) | 0.86 | ±0.01 |

## 5. Application to artificial domain

The first application domain we explore is an artificial "high entropy" clustering where cluster labels were generated using a uniformly random distribution, and where the ideal final clustering was known. In this domain, four hundred data points where randomly placed into 10 clusters to form an initial clustering. From this initial clustering, eight variants were created. Each variant was made by first duplicating the base clustering and then adding a perturbation by moving a random subset of the data points to new clusters (for each data point in the subset, the cluster to which it was moved was chosen independently over a uniform distribution over all the clusters). The amount of perturbation was specified by $B$, the fraction of all the data points contained in the subset. These new clusterings were used as the input to the ensemble clustering and experiments were performed with $B$ ranging from 0.0 to 1.0. The performance of the algorithms were determined by the NMI between the clustering produced by the algorithm and the initial clustering. Since this was an artificial data set we had "ground truth" about what the true clusterings should have been. A good cluster ensemble algorithm should produce a clustering as close as possible to the initial clustering from which the eight ensemble clusterings were derived.

### 5.1. Experimental set-up

In all the experiments, the VACs used a simple single-time-step reinforcement learner. At every time step, each VAC chose one of $k$ clusters based on its estimates of the reward for that choice. These estimates were stored in a table of size $k$. This process was done in an $\epsilon$-greedy fashion (with $\epsilon = 0.005$) where the highest valued cluster would be chosen with probability $1-\epsilon$ and a random cluster would be chosen with probability $\epsilon$. The learning rate $\alpha$ was set to 0.5.[4] After all the VACs chose their cluster, each VAC computed a reward (ANMI or DNMI) and used it to update its reinforcement learning policy. All experiments were performed with at least 30 trials, and we report the average values as well as the differences in the mean. In all cases, the performance of the system is given in terms of the global performance, measured by the NMI between the "true clustering" and the clustering we produced. For the artificial data the true clustering is known from the model. For the real world domains (presented in Section 6) the true clustering is determined from hand made assignments.

### 5.2. Results

In this experiment we tested the performance of the algorithms in the high entropy cluster domain. Tables 1–3 show the results for various values of $B$ (perturbation of the initial clusters). In this experiment, the VACs using *DNMI* performed similarly to the good graph-based methods and considerable better than the lowest per-

[4] Learning in this case refers to the change in the reward estimate for a particular vote and is given by $E^{\text{new}} \leftarrow \alpha R + (1-\alpha)E^{\text{old}}$ where $R$ is the immediate reward and $E^{\text{new}}$ and $E^{\text{old}}$ are the new and old estimates for the that vote's reward, respectively.

**Table 2**
Global performance (NMI) on artificial data ($B = 0.4$)

| Algorithm | Performance | Standard deviation |
|---|---|---|
| ***MCLA*** | ***0.91*** | – |
| ***CSPA*** | ***0.90*** | – |
| HPGA | 0.47 | – |
| G. OPT | 0.89 | – |
| ***VAC (DNMI)*** | ***0.90*** | ***±0.01*** |
| VAC (ANMI) | 0.27 | ±0.01 |

**Table 3**
Global performance (NMI) on artificial data ($B = 0.8$)

| Algorithm | Performance | Standard deviation |
|---|---|---|
| MCLA | 0.12 | – |
| CSPA | 0.17 | – |
| HPGA | 0.10 | – |
| ***G. OPT*** | ***0.18*** | – |
| VAC (DNMI) | 0.11 | ±0.003 |
| VAC (ANMI) | 0.07 | ±0.002 |

forming graph-based method, HGPA. In this domain, HGPA had particularly poor performance as compared to MCLA because MCLA could collapse similar edges and extract the true clusterings. In particular, for $B = 0$, the greedy optimizer, MCLA, and VACs using ANMI performed optimally. For $B = 0.4$ and $B = 0.8$, DNMI based VACs were again performing at the top or near the top (bold values show statistically significant "best" results in each table).

In addition to performing similarly to the best graph-based methods, VACs using DNMI performed equivalently to the simple greedy optimization method. However this performance was achieved with much less computation. The greedy optimization method was reported to take an hour on a 1 GHz PC (Strehl et al., 2002b). The VACs approach took only 45 s on a similar computer. In contrast to VACs using DNMI, VACs using ANMI directly performed very poorly. This poor performance can be attributed to the large number of VACs in this problem (80). When a VAC chose a cluster and the value of ANMI changed, the VAC did not know whether the change was caused by its vote or the vote of one of the seventy nine other VACs. This low performance of ANMI based VACs is likely the reason why active, learning-based cluster ensembles have not been pursued until now. The introduction of the DNMI is the first step in providing a viable, adaptive cluster ensemble method.

## 6. Real world application domain

In this section, we present results from two real world domain. First, we focus on clustering sets of hand-written images, where the true number of clusters was known, but the correct clustering was subject to interpretation. Then we look at a large document clustering problem (from the Yahoo! web site), where both the correct number of clusters and the correct clustering were subject to interpretation. The learning parameters as well as the experimental set-up is as described in Section 5.1, unless noted otherwise. We provide the results of the best graph-based method from the previous section for comparison for both domains (MCLA). The greedy optimization method however, was impractical in both real world applications due to the size of the data sets and the speed of convergence (recall, it took 80 times longer for greedy optimization to reach results of similar quality to VACs using DNMI in the simple artificial domain discussed in the previous section).

### 6.1. Handwriting recognition

In this domain, 1000 data points representing digits needed to be clustered. This was a difficult handwriting recognition problem, where clusters represented the digits, and each data point had 16 features (Alimoglu et al., 1997; Strehl et al., 2000). The data points were clustered into 10 clusters, with each cluster representing a digit from '0' to '9'. While this data set was known to have precisely 10 clusters, the true clustering was subjective since some of the images were ambiguous. From the original data set we produced eight different base clusterings by partitioning the data into eight data sets. Each point in the new data sets contained four features sampled from the original 16. Eight clusterings were then produced by clustering each of the new data sets with a graph partitioning algorithm (METIS).

The VACs using ANMI were able to perform better than the best centralized graph-based algorithm, MCLA (Table 4). VACs using ANMI directly did not perform well. This result shows again the value of using the DNMI reward, as clusters using this reward learned much more quickly and achieved much higher final performance. Using DNMI, a performance level equivalent to MCLA was achieved in only 25 learning steps.

In this experiment we also report results based on a certain percentage of VACs providing spurious votes. This is equivalent to VACs within particular clusterings being corrupted and voting for one cluster regardless of the data or the rewards (we provide all results with spurious votes in bold, as they provide new functionality not available to MCLA). This is a situation where graph-based methods no longer apply as their performance is based on the full system functioning properly. The results show that the VACs can overcome 10% spurious votes and provide good performance. In addition, they provide acceptable results even with 30% spurious votes.

### 6.2. Web documents

In the second real world domain, we analyzed web pages that naturally fell under different categories. In this domain the data set contained documents of varying categories from the Yahoo! Internet portal (Boley et al., 1999; Strehl et al., 2000). The data set contained 2340 data points, with each data point representing a text document through 2903 features (the features were a pruned set of word frequencies contained in the document). The data set was clustered into 20 clusters, based on the Yahoo! news category in which they were originally placed. Note that both the correct number of clusters and the true clusterings were subjective for this data set. The number of news categories could be modified, and many documents could be "correctly" assigned to more than one category.

We created 20 clusterings (coincidentally, each clustering having 20 clusters) by splitting the full data set into 20 separate data sets. Each new data set contained all of the 2340 data points, but each data point only had a 128 element subset of the original features. Each new data set was then clustered using a graph

**Table 4**
Global performance (NMI) for handwriting recognition

| Percent failures | Algorithm | Performance | Standard deviation |
|---|---|---|---|
| | MCLA | 0.58 | – |
| 0 | ***VAC (DNMI)*** | ***0.62*** | ***±0.004*** |
| | VAC (ANMI) | 0.35 | ± 0.01 |
| ***10*** | ***VAC (DNMI)*** | ***0.61*** | ***±0.005*** |
| ***30*** | ***VAC (DNMI)*** | ***0.44*** | ***±0.011*** |
| ***50*** | ***VAC (DNMI)*** | ***0.08*** | ***±0.02*** |

**Table 5**
Global performance (NMI) for web document clustering

| Percent Failures | Algorithm | Performance | Standard deviation |
|---|---|---|---|
| | ***MCLA*** | ***0.38*** | – |
| 0 | VAC (DNMI) | 0.34 | ±0.002 |
| | VAC (ANMI) | 0.08 | ±0.001 |
| ***10*** | ***VAC (DNMI)*** | ***0.32*** | ***±0.002*** |
| ***30*** | ***VAC (DNMI)*** | ***0.14*** | ***±0.007*** |
| ***50*** | ***VAC (DNMI)*** | ***0.08*** | ***±0.004*** |

partitioning algorithm (METIS), forming 20 clusterings. The performance of the VACs were compared to the original Yahoo! clustering using ANMI.

Table 5 shows the results. VACs using ANMI directly were unable to produce an adequate clustering. In fact, using ANMI directly performed even worse in this domain than in the handwriting recognition domain. This was not surprising since this data set required more VACs than the handwriting recognition data set, using a total of 400 VACs. In contrast VACs using *DNMI* were able to do significantly better. However VACs were not able to do as well as the MCLA algorithm.[5] This is however an interesting result in itself. This shows that VACs using DNMI are not necessarily superior to graph-based methods based on performance in all domains. However, the performance of VACs degraded gracefully (though in this case the drop of performance for 30% and 50% spurious votes was more severe). This allows the application of the proposed approach in many domains where gathering the data in a single location is impractical, and furthermore allows the use of ensemble clustering even when a given percentage of the VACs fail (e.g., provide spurious votes).

## 7. Discussion

Clustering plays a crucial part in many pattern recognition problems, particularly in problems with large data sets and large feature spaces. Clustering ensembles provide a natural way of achieving robust clusterings that are insensitive to noise in the data, and can be formed from distributed data centers. However, though extensively used for supervised classification algorithms, ensembles have not been widely used for non-supervised clustering algorithms. In this paper, we presented voting active clusters as a method for obtaining ensemble clusterings in a variety of domains.

Broadly, the results presented in this article show that the distributed, adaptive ensemble clustering methods performs comparably to the best existing centralized cluster ensemble methods under ideal conditions. VACs achieved a high level of performance with relatively little computation as compared to simple greedy optimization methods. In addition, the VAC method shows good fault tolerance properties and was able to provide acceptable performance even when up to 50% of the votes are spurious. This is a promising results, in that even in domains where centralized, graph-based methods provide better results than VACs in ideal conditions, the use of VACs is warranted if either the centralization assumption or the noise-free data assumption is violated.

While ANMI is a natural reward for this problem VACs attempting to optimize ANMI directly performed poorly. We surmise that this is one of the primary reasons for the dearth of adaptive cluster ensembles both in the clustering literature and the clustering toolboxes used in many applications. The introduction of the DNMI reward overcomes this problem, and we expect this advance to significantly increase the potential applications of distributed cluster ensemble algorithms.

## Acknowledgements

The authors would like to thank Joydeep Ghosh and Alexander Strehl for valuable discussions and their help in obtaining and interpreting the cluster ensemble results based on graph-based partitioning (MCLA, HPGA).

## References

Agogino, A., Tumer, K., 2004. Efficient evaluation functions for multi-rover systems. In: The Genetic and Evolutionary Computation Conference, Seatle, WA.
Agogino, A., Tumer, K., 2006. Efficient agent-based cluster ensembles. In: Proc. 5th Internat. Joint Conf. on Autonomous Agents and Multi-Agent Systems, Hakodate, Japan.
Alimoglu, F., Alpaydin, E., 1997. Combining multiple representations and classifiers for pen- based handwritten digit recognition. In: Fourth International Conference on Document Analysis and Recognition, vol. 2, Ulm, Germany.
Bishop, C.M., 1996. Neural Networks for Pattern Recognition. Oxford University Press.
Boley, D., Gini, M., Gross, R., Han, S., Hastings, K., Karypis, G., Kumar, V., Mobasher, B., Moore, J., 1999. Partitioning-based clustering for web document categorization. Decision Support Syst. 27 (3), 329–341.
Chakravarthy, S.V., Ghosh, J., 1996. Scale-based clustering using the radial basis function network. IEEE Trans. Neural Networks, 1250–1261.
Dhillon, I.S., Modha, D.S., 2001. Concept decompositions for large sparse text data using clustering. Mach. Learning 42 (1), 143–175.
Dietterich, T.G., 2000. Hierarchical reinforcement learning with the maxq value function decomposition. J. Artificial Intell. 13, 227–303.
Fern, X.Z., Brodley, C.E., 2004. Solving cluster ensemble problems by bipartite graph partitioning. In: ICML'04: Proc. 21st Internat. Conf. on Machine Learning. ACM Press, NY, USA.
Fowlkes, E.B., Mallows, C.L., 1983. A method for comparing two hierarchical clusterings. J. Amer. Statist. Assoc. 78 (383), 553–569.
Hubert, L., Arabie, P., 1985. Comparing partitions. J. Classification 2 (1), 193–218.
Kaelbing, L.P., Littman, M.L., Moore, A.W., 1996. Reinforcement learning: a survey. J. Artificial Intell. Res. 4, 237–285.
Karypis, G., Kumar, V., 1998. A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM J. Scientific Comput. 20 (1), 359–392.
Karypis, G., Aggarwal, R., Kumar, V., Shekhar, S., 1997. Multilevel hypergraph partitioning: applications in vlsi domain. In: Proc. Design and Automation Conference.
Kernighan, B.W., Lin, S., 1970. An efficient heuristic procedure for partitioning graphs. Bell Syst. Technical J. 42 (2), 291–307.
Kim, J., Warnow, T., 1999. Tutorial on phylogenetic tree estimation. In: Intell. Syst. Mol. Biol.. Heidelberg, Germany.
Kohonen, T., 1982. Self-organized formation of topologically correct feature maps. Biol. Cybernet. 43, 59–69.
MacQueen, J., 1967. Some methods for classification and analysis of multivariate observations. In: Proc. 5th Berkeley Symposium on Mathematical Statistics and Probability, vol. 1.
Merugu, S., Ghosh, J., 2003. A probabilistic approach to privacy-sensitive distributed data mining. In: Proc. 6th Internat. Conf. on Information Technology (CIT-2003), Bhubaneshwar.
Minaei-Bidgoli, B., Topchy, A., Punch, W., 2004. Ensembles of partitions via data resampling. In: Proc. IEEE Internat. Conf. on Information Technology: Coding and Computing, ITCC04.
Mitchell, T.M., 1997. Machine Learning. WCB/McGraw-Hill, Boston, MA.
Strehl, A., Ghosh, J., Mooney, R.J., 2000. Impact of similarity measures on web-page clustering. In: Proc. AAAI Workshop on AI for Web Search (AAAI 2000). Austin, AAAI/MIT Press.
Strehl, A., 2002. Relationship-based Clustering and Cluster Ensembles for High-Dimensional Data Mining, Ph.D. Thesis, The University of Texas at Austin.
Strehl, A., Ghosh, J., 2002. Cluster ensembles – a knowledge reuse framework for combining partitionings. In: Proc. AAAI 2002, Edmonton, Canada, AAAI.
Sutton, R.S., Barto, A.G., 1998. Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA.
Topchy, A., Jain, A.K., Punch, W., 2003. Combining multiple weak clusterings. In: ICDM'03: Proc. 3rd IEEE Internat. Conf. on Data Mining. IEEE Computer Society, Washington, DC, USA.

[5] Note that we could not generate the exact clusterings used in the papers (Strehl et al., 2002a,b), due to the randomness in the METIS algorithm. Our results for MCLA and the VACs were based on a set of clusterings that turned out to be slightly harder, therefore our results cannot be directly compared to the results in that paper.

Tumer, K., Wolpert, D. (Eds.), 2004. Collectives and the Design of Complex Systems. Springer, New York.
Tumer, K., Wolpert, D.H., 2000. Collective intelligence and Braess' paradox. In: Proc. 17th National Conf. on Artificial Intelligence, Austin, TX.
Witten, I., Frank, E., 2005. Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufman, San Francisco.
Wolpert, D.H., Tumer, K., 2001. Optimal payoff functions for members of collectives. Advances in Complex Systems 4 (2/3), 265–279.
Zhang, W., Dietterich, T.G., 1995. A Reinforcement Learning Approach to Job-shop Scheduling. Proceedings of the International Conference on Artificial Intelligence (IJCAI95).