Cluster Ensembles for High Dimensional Clustering: An Empirical Study

Xiaoli Z. Fern XZ@ECN.PURDUE.EDU
School of Electrical and Computer Engineering, Purdue University, W. Lafayette, IN 47907, USA
Carla E. Brodley BRODLEY@CS.TUFTS.EDU
Electrical Engineering and Computer Science, Tufts University, 161 College Avenue, Medford, MA 02155, USA

Editor:

Abstract

This paper studies cluster ensembles for high dimensional data clustering. We examine three different approaches to constructing cluster ensembles. To address high dimensionality, we focus on ensemble construction methods that build on two popular dimension reduction techniques, random projection and principal component analysis (PCA). We present evidence showing that ensembles generated by random projection perform better than those by PCA and further that this can be attributed to the capability of random projection to produce diverse base clusterings. We also examine four different consensus functions for combining the clusterings of the ensemble. We compare their performance using two types of ensembles, each with different properties. In both cases, we show that a recent consensus function based on bipartite graph partitioning achieves the best performance.

Keywords: Clustering, Cluster ensembles, Dimension reduction, Random projection

1. Introduction

Clustering for unsupervised data exploration and analysis has been investigated for decades in the statistics, data mining, and machine learning communities. The general goal of clustering is to partition a given set of data points in a multidimensional space into *clusters* such that the points within a cluster are similar to one another. Many clustering algorithms require a definition of a metric to compute the distance between data points. Thus, their performance are often directly influenced by the dimensionality used for calculating the chosen distance metric. Data sets with high dimensionality pose two fundamental challenges for clustering algorithms. First, in a high dimensional space the data tend to be sparse (the curse of dimensionality Bellman, 1961). Indeed, Beyer et al. (1999) demonstrated that as the dimensionality increases, the difference in distance between a given point and it nearest neighbor and other points in the data set often becomes negligible—making it difficult if not impossible to identify any clustering structure in the data based on distance measures. Second, there often exist irrelevant and/or noisy features that may mislead clustering algorithms (Dy and Brodley, 2004). For example, with the presence of irrelevant features, points from distinct clusters may appear to be closer (as measured by the Euclidean distance) than points from the same cluster.

To ameliorate the problems caused by high dimensionality, clustering is often used in combination with dimension reduction techniques. Representative examples of commonly used dimension reduction techniques include principle component analysis (PCA), feature selection (Agrawal et al., 1998, Dy and Brodley, 2004) and projection pursuit (Huber, 1985, Friedman, 1987). The goal of dimension reduction for clustering is to form a lower dimensional representation of the data such that "natural" clusters are easier for clustering algorithms to detect. To achieve this goal, dimension reduction techniques often rely on some "interestingness" criterion to search for a single lower dimensional representation. However, because the true structure of the data is unknown, it is inherently ambiguous what constitutes a good low dimensional representation. This makes it difficult to define a proper "interestingness" criterion. An alternative approach to this problem that we explore in this paper is to use multiple low-dimensional representations of the data. Each representation is used to cluster the data and a final clustering is obtained by combining all of the clustering solutions. Note that each low-dimensional representation presents the clustering algorithm with a different view of the original data. By combining the results from many different views, we expect to accumulate information about the structure of the data and produce better and more robust clustering results.

Although several recent papers have addressed cluster ensembles (Strehl and Ghosh, 2002, Fern and Brodley, 2003, Topchy et al., 2003), research in this area has been limited in a number of ways. First, there has not been a systematic investigation of this approach for high dimensional data clustering despite its clear promise as discussed above. Second, empirical evaluations of existing cluster ensemble systems have been limited to a small number of data sets, some of which are artificially constructed. We believe that a more thorough empirical evaluation is warranted. Lastly, the current cluster ensemble literature has mainly focused on the problem of combining multiple clusterings (commonly referred to as the problem of consensus function) and little guidance is available for how to construct cluster ensembles (i.e., produce multiple clustering solutions). Many different approaches can be used to construct cluster ensembles and they may produce ensembles that vary significantly with respect to the amount by which the individual clustering solutions differ from one another (diversity) and how good each clustering solution is (quality). Further, it is likely that different consensus functions may be better suited to different types of cluster ensembles. An in-depth study of the above issues will provide useful guidance for applying cluster ensemble techniques in practice.

This paper investigates cluster ensemble techniques in the context of high dimensional data clustering. First, we examine three approaches to constructing cluster ensembles. To address high dimensionality, our ensemble constructors build on two popular dimension reduction techniques: random projection (RP) and principle component analysis (PCA). Second, we examine four consensus functions for combining clusterings. Through an in-depth empirical analysis, our goal in this paper is to study:

- the suitability of different ensemble construction methods and different consensus functions for high dimensional clustering,

- the influence that the diversity and quality of base clusterings have on the final ensemble performance, and

- the relationship between the performance of various consensus functions and the basic properties (diversity and quality) of cluster ensembles.

Our study indicates that, in comparison to the traditional approach of a single clustering with PCA dimension reduction, cluster ensembles are generally more effective for high dimensional data clustering. Among the various ensemble construction techniques examined in this paper, the random projection based approach is shown to be most effective in producing ensembles with high diversity and good quality for high dimensional data. Last, our experiments on four different consensus functions show that although there is no universal single best consensus function, on average the bipartite graph partitioning consensus function (Fern and Brodley, 2004) is the most stable and effective approach for combining cluster ensembles.

The remainder of this paper is arranged as follows. In Section 2, we introduce the basics of cluster ensembles. Section 3 describes three different methods for generating cluster ensembles and Section 4 introduces different consensus functions. Section 5 describes the data sets used for our experiments, the parameter choices and experimental methods. Sections 6 and 7 present the experiments on ensemble constructors and consensus functions respectively. In Section 8, we review the related work on cluster ensembles. Section 9 summarizes the paper and discusses directions for future work.

2. Cluster Ensembles

A cluster ensemble consists of two parts: an *ensemble constructor* and a *consensus function*. Given a data set, an ensemble constructor generates a collection of clustering solutions—i.e., a cluster ensemble. A consensus function then combines the clustering solutions of the ensemble and produces a single clustering as the final output of the ensemble system. Below we formally describe these two parts.

Ensemble Constructor: Given a data set of n instances $X = \{X_1, X_2, \cdots, X_n\}$, an ensemble constructor generates a *cluster ensemble*, represented as $\Pi = \{\pi^1, \pi^2, \cdots, \pi^r\}$, where r is the ensemble size (the number of clusterings in the ensemble). Each clustering solution π^i is simply a partition of the data set X into K^i disjoint clusters of instances, represented as $\pi^i = \{c_1^i, c_2^i, \cdots, c_{K^i}^i\}$, where $\cup_k c_k^i = X$. For simplicity this paper assumes that hard clusterings are used. However, it should be noted that each consensus function examined in this paper can be applied to cluster ensembles with soft clusterings, directly or with minor modifications.

Consensus Function: Given a cluster ensemble Π and a number K, the desired number of clusters, a consensus function Γ uses the information provided by Π and partitions X into K disjoint clusters as the final clustering solution π_f. In a more general case, one can also use the original features of X in combination with Π to produce the final clustering. In this study we focus on the case where the original features of X are only used during ensemble construction.

3. Ensemble Constructors

In this section, we introduce three ensemble constructors. Because our goal in this paper is to cluster high dimensional data, we focus on approaches that build on dimension reduction techniques. Note that for all approaches described below, we use K-means as the base clustering algorithm for its simplicity and computational efficiency.

3.1 Random Projection Based Approach

Our first ensemble constructor is based on random projection, a dimension reduction technique that has seen growing popularity due to its simplicity and theoretical promise. We will refer to this approach as the RP-based approach and the resulting ensembles as RP ensembles.

Our motivation for using random projection is threefold. First, in comparison to traditional dimension reduction methods such as PCA, random projection is a generic dimension reduction technique that does not use any "interestingness" criterion to "optimize" the projection. Specifying a proper "interestingness" criterion for a data set can be challenging due to the unsupervised nature of clustering tasks. Second, random projection has been shown to have promise for high dimensional data clustering. In 1984, Diaconis and Freedman showed that various high-dimensional distributions look more Gaussian when randomly projected onto a low-dimensional subspace. Recently, Dasgupta (2000) showed that random projection can change the shape of highly eccentric clusters to be more spherical. These results suggest that random projection combined with standard K-means clustering may be well suited to finding structure in high dimensional data. Lastly, our initial experiments (Fern and Brodley, 2003) on random projection for clustering high dimensional data indicate that clustering with different random projection runs results in different clustering solutions that can be complementary to each other. We conjecture that this is a desirable property for cluster ensembles, rendering random projection a natural candidate for constructing cluster ensembles. Below we introduce random projection and the RP-based ensemble constructor.

A random projection from d dimensions to d' dimensions is simply a randomly generated linear transformation, represented by a $d \times d'$ matrix R. The transformation matrix R is typically generated by first setting each entry of the matrix to a value drawn from an i.i.d. N(0,1) distribution and then normalizing the columns to unit length. Generally speaking, a matrix R generated as described above will not be orthonormal. To obtain a real projection matrix, we need to further orthogonalize R. However, as shown by Hecht-Nielsen (1994), high dimensional vectors with random directions may be sufficiently close to being orthogonal. Therefore, in our experiments we do not perform orthogonalization.

Given a d-dimensional data set represented as an $n \times d$ matrix X, where n is the number of total data points in X, the mapping $X \times R$ results in a reduced-dimension data set X'. We then apply the standard K-means algorithm to the new data set X' to obtain a clustering solution. To construct a cluster ensemble, the above process is repeated multiple times and each time a different projection matrix is randomly generated.

The RP-based ensemble constructor requires two parameters: d', the new dimension for random projection; and k_0, the desired number of clusters for K-means in each clustering run. We will discuss the parameter choices in Section 5.2.

3.2 Combining PCA and Random Subsampling

PCA is a widely used dimension reduction technique. It selects projection directions to maximize data variance. Although the variance criterion used by PCA cannot guarantee finding a good low-dimensional representation of the data for clustering (Fukunaga, 1990), it is often considered to be a useful heuristic. Indeed, when comparing random projection with PCA applied to produce a *single* clustering, we observed that in many cases PCA leads to better performance than random projection. Given a data set, PCA chooses projection directions deterministically, therefore, we combine PCA with subsampling to construct cluster ensembles.

Specifically, given a high-dimensional data set X, we first apply PCA to reduce the dimension of the data from from d to d'. Random subsampling with a sampling rate of 65% [1] is then applied to generate multiple subsamples of the reduced data. To avoid duplicating instances, the subsampling is performed without replacement. Each subsample is then clustered by K-means. Note that K-means only clusters instances that appear in the current subsample—resulting in the possibility that some instances are never clustered in the entire ensemble. To avoid this situation, in each clustering run we further assign each instance that is absent from the current subsample to its closest cluster based on its Euclidean distances (using the reduced d'-dimensional representation) to the cluster centers.

In the remaining part of the paper, we will refer to this ensemble constructor as PCASS and the resulting ensembles as PCASS ensembles. Note that we also explored an alternative approach, where we subsample first and then apply PCA. By doing so, subsampling is used as a tool for producing different principle projection directions. However, empirical comparisons between these two approaches indicate that they produce highly similar cluster ensembles. An possible explanation is that the features of the high dimensional data sets studied are highly redundant—applying PCA to subsampled data leads to very similar projection directions. In light of the similar behavior of the two approaches, we will focus on PCASS in our future discussion.

3.3 Combining Random Projection with PCA

When PCA is used in PCASS for constructing cluster ensembles, a critical limitation is that it produces only a single low-dimensional representation of the data in different clustering runs. If significant information is erroneously discarded by PCA, all clustering runs (performed on subsamples) in the ensemble will suffer the same loss without exception. In contrast, although a single run of random projection may lead to significant information loss, the other random projection runs in the ensemble can potentially compensate for the loss incurred in that run. To benefit from the strengths of both approaches, our third ensemble constructor combines random projection with PCA.

Given a high-dimensional data set X, we first apply random projection to reduce the dimension of the data set from d to d_1. PCA is then applied to the d_1-dimensional data to further reduce the dimension to d_2 and form the final low-dimensional representation of the data, which is then clustered by K-means. We refer to this ensemble constructor as RPPCA, and the resulting ensembles as RPPCA ensembles.

[1] The sampling rate was arbitrarily selected.

4. Consensus Functions

A consensus function takes an ensemble of clustering solutions and combines them to produce a single (presumably better) final clustering. Note that combining clusterings is a difficult task that does not come with a simple and natural solution. Because there is no clear correspondence relationship among clusters obtained from different clusterings, we can not directly apply simple schemes such as majority vote. In this paper we examine four different consensus functions, each of which approaches the clustering combination problem by transforming it into a more familiar problem. In particular, the first three consensus functions each reduce the clustering combination problem to a graph partitioning problem, and the fourth approach reduces it to a standard clustering problem.

4.1 Consensus Functions Using Graph Partitioning

Consensus functions using graph partitioning have two main advantages. First, graph partitioning is a well studied area and algorithms such as spectral clustering have been successful in a variety of applications (Shi and Malik, 2000, Dhillon, 2001). Second, cluster ensembles provide a simple and effective way to define similarity measures for computing the weight of the edges in a graph, which is an important and sometimes hard to satisfy prerequisite for the success of graph partitioning techniques (Bach and Jordan, 2003). Below we first describe the basics of graph partitioning and provide the necessary notation.

The input to a graph partitioning problem is a graph that consists of vertices and weighted edges, represented by $G = (V, W)$, where V is a set of vertices and W is a nonnegative and symmetric $|V| \times |V|$ similarity matrix characterizing the similarity between each pair of vertices. To partition a graph into K parts is to find K disjoint sets of vertices $P = \{P_1, P_2, \cdots, P_K\}$, where $\cup_k P_k = V$. Unless a given graph has K, or more than K, strongly connected components, any K-way partition will cross some of the graph edges. The sum of the weights of these crossed edges is often referred to as the cut of a partition P: $Cut(P, W) = \sum W(i, j)$, where vertices i and j do not belong to the same set.

Given a weighted graph, a general goal of graph partitioning is to find a K-way partition that minimizes the cut, subject to the constraint that each part should contain roughly the same number of vertices. In practice, various graph partitioning algorithms define different optimization criteria based on the above goal. In this paper, we use a well-known spectral graph partitioning algorithm by Ng et al. (2002).

Given the basics of graph partitioning, we now introduce three different consensus functions, each of which formulates and solves a different graph partitioning problem given a cluster ensemble. The first approach models instances as vertices in a graph and the second models clusters as vertices. They were proposed by Strehl and Ghosh (2002). Here we refer to them as the instance-based and cluster-based graph formulations respectively to characterize their key distinction. Finally, the third consensus function, recently proposed by Fern and Brodley (2004), models instances and clusters simultaneously as vertices in a bipartite graph.

4.1.1 Instance-Based Graph Formulation

The Instance-Based Graph Formulation (IBGF) approach constructs a graph that models the pairwise relationship among instances of the data set X. Below we formally describe IBGF.

Given a cluster ensemble $\Pi = \{\pi^1, \cdots, \pi^R\}$, IBGF constructs a fully connected graph $G = (V, W)$, where

- V is a set of n vertices, each representing an instance of X.
- W is a similarity matrix and $W(i,j) = \frac{1}{R}\sum_{r=1}^{R} I(g_r(X_i) = g_r(X_j))$, where $I(\cdot)$ is an indicator function that returns 1 if the argument is true and 0 otherwise; $g_r(\cdot)$ takes an instance and returns the cluster to which it belongs in π^r.

$W(i,j)$ measures how frequently the instances i and j are clustered together in the given ensemble. In recent work (Fern and Brodley, 2003, Monti et al., 2003), this similarity measure has been shown to give satisfactory performance in domains where a good similarity (or distance) metric is otherwise hard to find. Once a graph is constructed, one can solve the graph partitioning problem using any chosen graph partitioning algorithm and the resulting partition can be directly output as the final clustering solution.

Note that IBGF constructs a fully connected graph with n^2 edges, where n is the number of instances. Depending on the algorithm used to partition the graph, the computational complexity of IBGF may vary. But generally it is computationally more expensive than the other two graph formulation approaches examined in this paper.

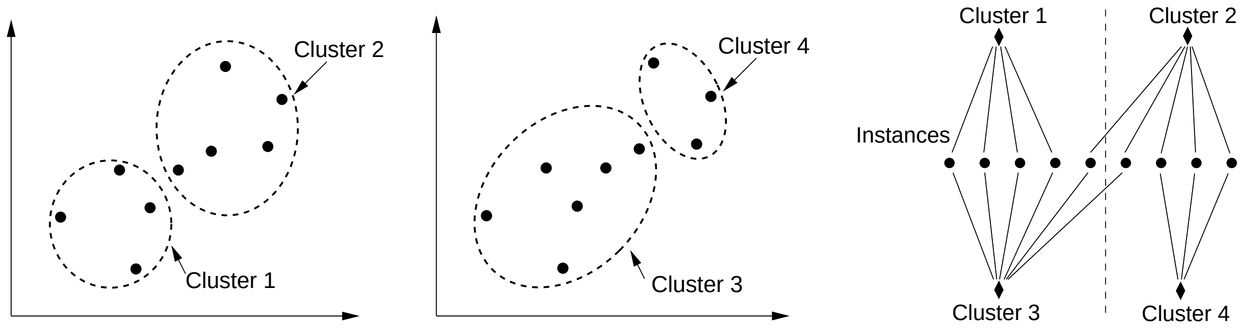
4.1.2 Cluster-Based Graph Formulation

Note that clusters formed in different clusterings may contain the same set of instances or largely overlap with each other. Such clusters are considered to be corresponding (similar) to one another. Cluster-Based Graph Formulation (CBGF) constructs a graph that models the correspondence (similarity) relationship among different clusters in a given ensemble and partitions the graph into groups such that the clusters of the same group correspond (are similar) to one another.

Given a cluster ensemble $\Pi = \{\pi^1, \cdots, \pi^R\}$, we first rewrite Π as $\Pi = \{c_1^1, \cdots, c_{K_1}^1, \cdots, c_1^R, \cdots, c_{K_R}^R\}$ where c_j^i represents the jth cluster formed in the ith clustering π^i in the ensemble Π. Denote the total number of clusters in Π as $t = \sum_{r=1}^{R} K_r$. CBGF constructs a graph $G = (V, W)$, where

- V is a set of t vertices, each representing a cluster
- W is a matrix such that $W(i,j)$ is the similarity between the clusters c_i and c_j and is computed using the Jaccard measure as: $W(i,j) = \frac{|c_i \cap c_j|}{|c_i \cup c_j|}$

Partitioning a cluster-based graph results in a grouping of the clusters. We then produce a final clustering of instances as follows. First we consider each group of clusters as a metacluster. For each clustering in the ensemble, an instance is considered to be *associated with* a metacluster if it contains the cluster to which the instance belongs. Note that an instance may be associated with different metaclusters in different runs. We assign an



(a) Clustering 1 (b) Clustering 2 (c) Bipartite Graph

Figure 1: An example of the Hybrid Bipartite Graph Formulation

instance to the metacluster with which it is most frequently associated. Ties are broken randomly.

A basic assumption of CBGF is that there exists a correspondence structure among different clusters formed in the ensemble. This poses a potential problem—in cases where no such correspondence structure exists, this approach may fail to provide satisfactory performance. The advantage of CBGF is that it is computationally efficient. It generates a fully connected graph with t^2 edges, where t is the total number of clusters in the ensemble. This is significantly smaller than the n^2 of IBGF, assuming $t \ll n$.

Strehl and Ghosh (2002) also proposed a hypergraph-based approach, which models clusters as hyperedges and instances as vertices in a hypergraph and uses a hypergraph partitioning algorithm to produce a final partition. Conceptually, this approach forms a different type of graph and has the limitation that it can not model soft clustering. Practically, our initial experiments show that it performed worse than IBGF and CBGF on our data sets. Due to the above reasons, we choose not to further discuss this approach in this paper.

4.1.3 Hybrid Bipartite Graph Formulation

The Hybrid Bipartite Graph Formulation (HBGF) approach models instances and clusters simultaneously in a graph. The graph edges can only connect instance vertices to cluster vertices, resulting a bipartite graph. Below we first describe HBGF and then explain its conceptual advantages over IBGF and CBGF.

Description of HBGF: Given a cluster ensemble $\Pi = \{\pi^1, \cdots, \pi^R\}$, HBGF constructs a graph $G = (V, W)$, where

- $V = V^C \cup V^I$, where V^C contains t vertices each representing a cluster of the ensemble; V^I contains n vertices each representing an instance of the data set X.

- W is defined as follows. If the vertices i and j are both clusters or both instances, $W(i, j) = 0$; otherwise if instance i belongs to cluster j, $W(i, j) = W(j, i) = 1$ and 0 otherwise.

Note that W can be written as: $W = \begin{bmatrix} 0 & A^T \\ A & 0 \end{bmatrix}$ where A is a connectivity matrix whose rows correspond to the instances and columns correspond to the clusters. $A(i, j)$ is an indicator that takes value 1 if instance i belongs to the j-th cluster and 0 otherwise.

Figure 1 shows an example of HBGF. Particularly, Figures 1(a) and (b) depict two different clusterings of nine instances and Figure 1(c) shows the graph constructed by HBGF, in which the diamond vertices represent the clusters and the round vertices represent the instances. An edge between an instance vertex and a cluster vertex indicates that the cluster contains the instance. All edges in the graph have equal weight of one (edges with zero weights are omitted from the graph). In this graph, cluster vertices are only connected to instance vertices and vice versa, forming a bipartite graph. If a new clustering is added to the ensemble, a new set of cluster vertices will be added to the graph and each cluster vertex will be connected to the instances that it contains.

Shown in Figure 1(c) as a dashed line, a partition of the bipartite graph partitions the cluster vertices and the instance vertices simultaneously. The partition of the instances can then be output as the final clustering.

Although HBGF's vertex set is the sum of IBGF's and CBGF's vertex sets, it forms a sparse graph and, as shown by Dhillon (2001), due to its special structure the real size of the resulting bipartite graph partitioning problem is $n \times t$, where n is the number of instances and t is the total number of clusters in the ensemble C. This is significantly smaller compared to the size n^2 of IBGF, assuming that $t \ll n$.

Conceptual advantages of HBGF: Comparing to IBGF and CBGF, HBGF offers two important conceptual advantages. First, the reduction of HBGF is lossless — the original cluster ensemble can be easily reconstructed from an HBGF graph. In contrast, IBGF and CBGF do not have this property. To understand the second advantage of HBGF, it should be noted that IBGF and CBGF consider the similarity of instances and the similarity of clusters independently and, as shown below, such independent treatment may be problematic.

Comparing two pairs of instances (X_1, X_2) and (X_3, X_4), we assume that X_1 and X_2 are never clustered together in the ensemble and the same is true for pair (X_3, X_4). However, the instances X_1 and X_2 are each frequently clustered together with the same group of instances in the ensemble, i.e., X_1 and X_2 are frequently assigned to clusters that are similar to each other. In contrast, this is not true for X_3 and X_4. Intuitively we consider X_1 and X_2 to be more similar to one another than X_3 and X_4. However, IBGF will fail to differentiate these two cases and assign both similarities to be zero. This is because IBGF ignores the information about the similarity of clusters while computing the similarity of instances.

A similar problem exists for CBGF. For example, consider two pairs of clusters (c_1, c_2) and (c_3, c_4). Assume that $c_1 \cap c_2 = \phi$ and $c_3 \cap c_4 = \phi$. And further assume that the instances of c_1 and those of c_2 are often clustered together in other clustering runs, whereas this is not the case for c_3 and c_4. Note that CBGF will assign both similarities to zero while intuitively we would consider clusters c_1 and c_2 to be more similar to one another than clusters c_3 and c_4. CBGF fails to differentiate these two cases, because it does not take the similarity of instances into account.

Table 1: Description of the data sets

Name	Description	Source
EOS	Land-cover classification data set #1	Friedl et al. (2002)
MODIS	Land-cover classification data set #2	
HRCT	High resolution computed tomography lung image data	Dy et al. (1999)
CHART	Synthetically generated control chart time series	UCI KDD archive (Hettich and Bay, 1999)
ISOLET6	Spoken letter recognition data set (6 letters only)	UCI ML archive (Blake and Merz, 1998)
MFEAT	Handwritten digits represented by Fourier coefficients	
SATIMAGE	StatLog Satellite image data set (training set)	
SEGMENTATION	Image segmentation data	

In contrast, HBGF allows the similarity of instances and the similarity of clusters to be considered simultaneously in producing the final clustering. Thus it avoids the above problems of IBGF and CBGF.

4.2 Consensus Function Using Centroid-based Clustering

The fourth consensus function, proposed by Topchy et al. (2003), transforms a cluster ensemble problem into a standard clustering problem by representing a given ensemble as a new set of features and then uses the standard K-means clustering algorithm to produce the final clustering. Below we refer to this consensus function as KMCF, standing for K-Means Consensus Function.

Given a cluster ensemble $\Pi = \{\pi^1, \cdots, \pi^r\}$, KMCF first creates a new set of features as follows. For each partition $\pi^i = \{c^i_1, c^i_2, \cdots, c^i_{K^i}\}$, it adds K^i binary features to the new feature set, each corresponding to a cluster in π^i. The resulting new feature set will contain t features, where t is the total number of clusters of ensemble Π. Note that for an instance, the feature corresponding to c^i_j takes value 1 if that instance belongs to cluster c^i_j, and 0 otherwise. The new features are then standardized to have zero mean. Finally, K-means is applied to the standardized new features to produce the final clustering.

5. Experimental Setup

In this section we describe the data sets used in our experiments, related parameter settings and the evaluation criterion.

Table 2: Summary of the data sets

	CHART	EOS	HRCT	ISOL.	MFEAT	MODIS	SATI.	SEGM.
#INST.	600	2398	1545	1800	2000	4975	4435	2310
#CLASS	6	8	8	6	10	10	6	7
ORG. DIM.	60	20	183	617	76	112	36	19
d_r, d_p, d_2	10	5	20	30	10	10	5	5
d_1	20	10	40	60	20	20	10	10
k_0	10	15	15	15	15	15	15	15

Table 3: Required parameters for ensemble constructors.

Approach	Parameters	
RP	k_0	# of clusters for each clustering run
	d_r	dimension to reduce to for RP
PCASS	k_0	# of clusters for each clustering run
	d_p	dimension to reduce to for PCA
	r	subsampling rate
RPPCA	k_0	# of clusters for each clustering run
	d_1	Intermedium dimension for RP
	d_2	Final dimension for PCA

5.1 Description of the Data Sets

Our experiments use eight data sets obtained from a variety of domains and sources. Table 1 briefly describes these data sets. Note that the ISOLET6 data set is a subsample of the UCI Spoken letter recognition data set obtained by randomly selecting six out of a total of twenty-six letters. We chose to use only six letters to make sure that the IBGF approach can be applied within a reasonable amount of time. The MFEAT data set comes from the UCI multiple-feature data set. Note that the original data set contains several different representations of the same data, among which the fourier coefficients were used in our experiments. In rows 2-4 of Table 2 we present the main characteristics of these data sets. The dimensions of these data sets range from dozens to hundreds. Rows 5-7 list the parameters selected for each data set, which we discuss in Section 5.2. It should be noted that all of the eight data sets are labeled. However, the labels are omitted during clustering, and used only for evaluation.

5.2 Parameter Settings

In this paper, we examine three different ensemble constructors and four consensus functions. For the consensus functions, the only parameter to be specified is k_f, the desired number of the final clusters. Because we have access to the class information of our data sets, our experiments set k_f to be the same as the class number (shown in row 3 of Table 2) for all data sets and all consensus functions.

In Table 3, we summarize the parameters to be specified for each ensemble constructor. One parameter that is common to all ensemble constructors in k_0, the number of clusters for K-means in each clustering run during ensemble construction. The choices of k_0 for each data set are shown in the last row of Table 3. These numbers are arbitrarily selected. Note that we use a slightly smaller number for the CHART data set because it contains only 600 instances.

The remaining parameters are decided as follows. First, for PCASS, we select d_p, the dimensionality for PCA, by requiring that 90% of the data variance be preserved. Note that because the resulting dimensionalities for HRCT, ISOLET6 and MFEAT are still high, for computation efficiency we relax the requirements for these three data sets to preserving only 80% of the data variance. The sampling rate r of PCASS is set to be 65% (arbitrarily selected). In order to have direct comparisons between RP and PCASS, we set d_r, the dimensionality for RP, to be the same as d_p. Finally, for RPPCA, we have two dimension parameters, the intermediate dimension d_1 used by RP and the final dimension d_2 used by PCA. We set the final dimension d_2 to be the same as d_p and set d_1 to be $2 \times d_p$.

Here we argue that the exact choices of the above discussed parameters will not strongly influence the final performance. While the quality of the individual base clusterings may be significantly affected when different parameters are used, as we combine more and more base clusterings, we expect the ensemble performance to be comparable for different parameter settings. To verify our conjecture, we examined other choices for both d_p (the final dimensionality used by ensemble constructors) and k_0 (the cluster number for K-means in base clustering). Indeed, we also examined the case where we select d_p and k_0 randomly for each individual clustering run from a given range. All settings result in comparable performance. This is possibly due to the fact that cluster ensembles aggregate the information of multiple base clusterings—although the individual base clusterings may differ when parameters change, the total aggregated information remains comparable.

5.3 Evaluation Criterion

Because the data sets used in our experiments are labeled, we can assess the quality of the final clusterings using external criteria that measure the discrepancy between the structure defined by a clustering and what is defined by the class labels. Here we choose to use an information theoretic criterion—the Normalized Mutual Information (NMI) criterion (Strehl and Ghosh, 2002). Treating cluster labels and class labels as random variables, NMI measures the mutual information (Cover and Thomas, 1991) shared by the two random variables and normalizes it to a $[0, 1]$ range. Note that the expected NMI value of a random partition of the data is 0 and the optimal value 1 is attained when the class labels and cluster labels define the same partition of the data.

6. Experiments on Ensemble Constructors

In this section, we empirically evaluate the three ensemble constructors described in Section 3. The purpose of our experiments is to study the suitability of the various ensemble constructors for the high dimensional clustering task.

6.1 Upper-Bound Performance Analysis

Given a cluster ensemble we have a variety of choices for consensus functions and for an ensemble using a different consensus function may lead to a different result. To obtain a unified evaluation of the ensemble constructors, we evaluate each cluster ensemble using the following procedure. Given a cluster ensemble, we apply all of the four consensus functions to the ensemble, resulting in four final clusterings. We then evaluate the obtained four clusterings using the class labels and record the best NMI value obtained. We refer to this as the upper-bound performance evaluation for each ensemble constructor.

In Figure 2, we plot the upper-bound performance of the three ensemble constructors. The y-axis shows the NMI value and the x-axis shows the ensemble size. Each performance line consists of eleven data points. The first left-most point has ensemble size one, showing the average quality of the base clusterings produced by an ensemble constructor. To obtain this value, we use each ensemble constructor to generate an ensemble of size 50 and average the quality of the 50 base clusterings. The other ten data points show the performance of ensembles with ten different ensemble sizes, ranging from 10 to 100. For each ensemble size, ten different ensembles are generated using each constructor and the plotted values are averaged across the ten runs.

Comparing with base clusterings: Recall that the optimal NMI value is 1, which is achieved when the evaluated clustering and the class labels define the exact same partition of the data. The larger the NMI value the better the performance. Comparing the ensembles with their base clusterings, from Figure 2 we see that RP ensembles consistently improve over the base clusterings for all eight data sets. In contrast, RPPCA fails to improve for the SEGMENTATION data set, whereas PCASS fails to improve for the HRCT, EOS and MODIS data sets.

Comparing with regular clustering with PCA dimension reduction: As another base-line comparison, we apply PCA to each data set to reduce the dimension to d_p (using the same values as shown in Table 2), and then apply K-means to the reduced data with k equal to the class number. The resulting performance is shown in Figure 2 by the dotted lines. [2] As shown by the figures, all three types of ensembles outperform regular clustering with PCA dimension reduction.

Comparing the three ensemble constructors: From Figure 2, we observe that RP ensembles achieve the best performance among the three ensemble constructors for all but the SATIMAGE data set. The performance lines of RP ensembles generally show a clear improving trend as the ensemble size increases. RPPCA ensembles also improve with an

[2] Note that it may appear surprising that in some cases this base-line performance is inferior to the performance of the base clusterings. This can be explained by the fact that larger k values are used by K-means in producing base clusterings.

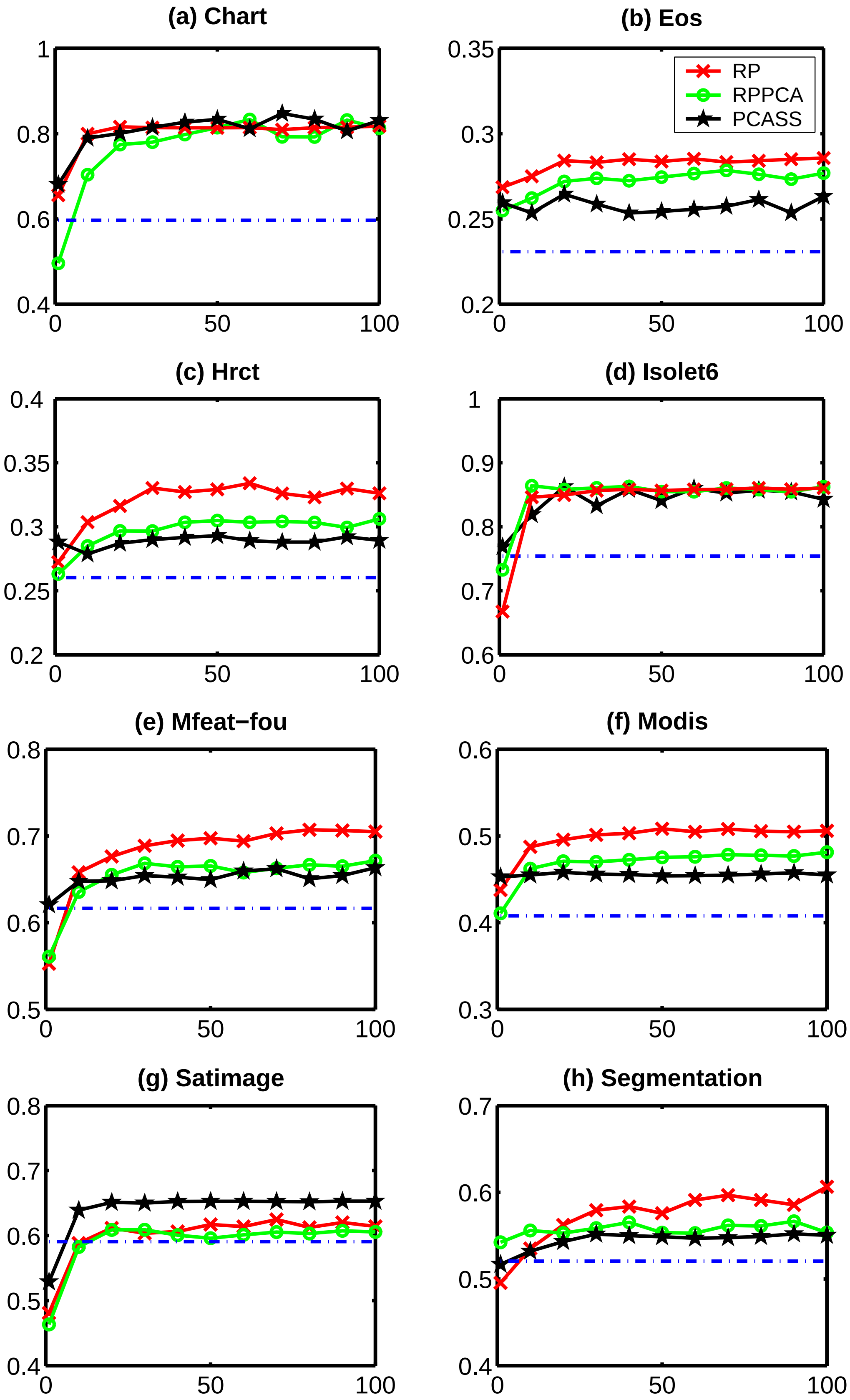


Figure 2: Upper-bound performance of the three types of ensembles.

increase in ensemble size, but its final performance tends to be inferior to RP. In contrast, PCASS behaves rather differently from the other two. Particularly, its base clusterings often have better quality compared to the other two approaches while the performance improvements introduced by ensembles are often less significant or even negligible in some cases. Interestingly, we notice that for the ISOLET6 and MFEAT data sets, the base clusterings generated by RP have significantly lower quality compared to those by PCASS. However, as we increase the ensemble size, RP ensembles gradually improve to have similar (for ISOLET6) or better (for MFEAT) final performance.

The above comparisons suggest that, among the three ensemble constructors, random projection (RP) generates the best performing cluster ensembles in most cases. Further, we conjecture that the superiority of RP ensembles is because RP produces more diverse base clusterings. To verify this conjecture, we examine the diversity and quality of the cluster ensembles using the process described below. Note that the approach we take here is similar to what was used by Dietterich (2000) for analyzing supervised ensembles.

Diversity-quality analysis: Given a cluster ensemble, we graph the diversity versus quality for each pair of clusterings in the ensemble as a cloud of diversity-quality (d-q) points. Each d-q point corresponds to a pair of base clusterings in the ensemble. To measure the diversity (shown on the y axis of a graph), we calculate the NMI *between each pair of base clusterings.* To obtain a single quality measure (shown on the x axis of a graph) for each pair, we average their NMI values as computed between each clustering and the class labels. Note that when the NMI *between* two clusterings is zero the diversity is maximized. In contrast, maximizing the *average* NMI of each pair maximizes their quality. From the distribution of the d-q points, we can obtain information about the quality and diversity of a given ensemble. For example, for an ensemble to have high diversity and good quality, its d-q points should be close to the right-bottom region of a graph.

We will first compare the RP ensembles and the PCASS ensembles in Figure 3 by examining the distribution of the d-q points of both approaches. For each approach, we obtain its d-q points using an ensemble of size 50. Note that the d-q points shown in the figures are uniformly subsampled to enhance the readability. Our first observation is that, for all eight data sets, the d-q points of RP ensembles are located below those of PCASS ensembles, indicating that the clusterings of RP ensembles are generally more diverse than those of PCASS. Further, we relate the results shown in Figure 3 to the performance evaluation shown in Figure 2. Examining each data set individually, we make the following observations.

- For the CHART data set, the d-q points of RP ensembles and PCASS ensembles largely overlap. Correspondingly, we see that the performance lines of these two ensembles as shown in Figure 2(a) are also close to each other.

- For the EOS, HRCT, MODIS, and SEGMENTATION data sets, we see that the quality of the two types of ensembles are similar whereas the diversity of RP ensembles are higher. Comparing the upper-bound performance of the two approaches for these four data sets shown in Figure 2, we observe that higher diversity leads to larger performance improvement over the base clusterings.

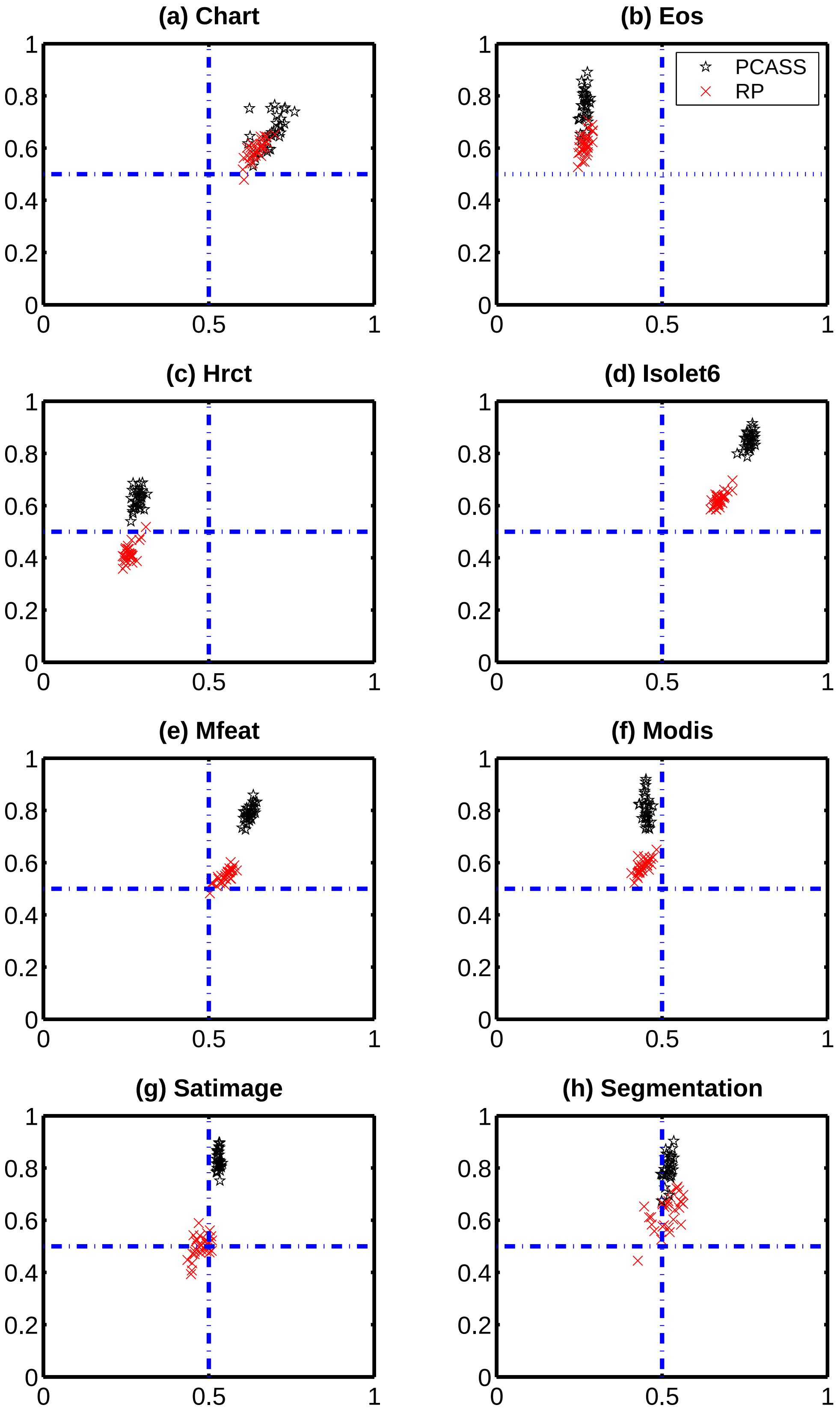


Figure 3: Diversity-quality analysis of RP ensembles and PCASS ensembles.

Table 4: Diversity and quality of RP ensembles and RPPCA ensembles.

	RP		RPPCA	
	Diversity	Quality	Diversity	Quality
CHART	0.588	0.644	0.454	0.502
EOS	0.624	0.269	0.647	0.255
HRCT	0.431	0.275	0.434	0.260
SATIMAGE	0.496	0.483	0.517	0.461
MODIS	0.596	0.442	0.588	0.405
ISOLET6	0.608	0.667	0.740	0.739
MFEAT	0.546	0.548	0.596	0.552
SEGMENTATION	0.607	0.478	0.736	0.535

- For the ISOLET6, MFEAT and SATIMAGE data sets, we note that the RP ensembles have lower quality but higher diversity compared to the PCASS ensembles. For ISOLET6 and MFEAT, we observe that, in comparison to PCASS, the final performance of RP is comparable or better although it begins with base clusterings of lower quality. This suggests that, for cluster ensembles, lower quality in the base clusterings may be compensated for by higher diversity. However, this is not always the case. Particularly, we also observe that PCASS outperforms RP for the SATIAMGE data set. The difference between SATIMAGE and the other two data sets is that SATIMAGE is the only data set where RP's d-q points lie in a different region from PCASS's d-q points. Particularly, all of the PCASS's d-q points have quality higher than 0.5, whereas many of RP's points have quality lower than 0.5.

In comparing RPPCA with RP, it is rather surprising to see that RPPCA performs so poorly compared to RP because RPPCA was designed with the goal of combining the strength of random projection and PCA. Indeed, our empirical evaluation appears to suggest that RPPCA actually inherits the weakness of RP and PCA. To analyze the diversity and quality of RP ensembles and RPPCA ensembles, we show the average diversity and quality of the two types of ensembles in Table 4. We present this comparison using a table because the d-q points of RPPCA tend to overlap those of RP, making it hard to interpret a figure.

In Table 4, we present the eight data sets in two groups based on whether RPPCA generates base clusterings of higher quality compared to RP. The first group contains five data sets. For these five data sets, RPPCA actually generates base clusterings that are of lower quality than those generated by RP. This indicates that our approach of combining PCA with RP does not necessarily lead to better dimension reduction performance for clustering, at least with the parameters setting used in this paper. Among these five data sets, the diversity of RPPCA ensembles is significantly higher than RP only for the CHART data set, for which the final ensemble performance of RP and RPPCA are comparable. For the other four data sets in the first group, the diversity of RPPCA ensembles are either lower (EOS, SATIMAGE) or similar (HRCT, MODIS). Correspondingly, we observe that RPPCA ensembles perform worse than RP ensembles for these four data sets. In the second group, we see three data sets for which RPPCA does improve the base clusterings compared to RP. However, as we see from the table, RP ensembles are much more diverse for these three data sets. Consequently, the final performance of RP ensembles are still better or at least similar compared to RPPCA ensembles.

In conclusion, the above diversity-quality analysis shows strong evidence that the high diversity of RP ensembles is, if not the direct reason, at least an important contributing factor for its superior performance in comparison to the other two approaches. Generally speaking, our observation suggests that if the base clusterings of two ensembles are of similar quality, the higher the diversity the better the final ensemble performance. In cases where the base clusterings of two ensembles differ in quality, high diversity may also compensate for low quality. But this may depend on the particular data set and the extend to which the quality drops.

6.2 Reality Performance Analysis

Note that the upper-bound evaluation used in the previous section measures the best performance achievable by each ensemble constructor using the four consensus functions described in Section 4. In reality, we can not achieve this performance because we do not have the ground truth for selecting the best result from four candidate clustering outputs. To evaluate the ensemble constructors in a more realistic setting, we use a second procedure based on an objective function defined by Strehl and Ghosh (2002). This objective function, referred to as the Sum of NMI (SNMI), measures the total amount of information shared by a final clustering and the participating clusterings in a given ensemble and is computed as follows. Given a cluster ensemble $\Pi = \{\pi^1, \cdots, \pi^r\}$ and a final cluster π_f, SNMI is defined as:

$$SNMI = \sum_{i=1}^{r} NMI(\pi^i, \pi^f)$$

where $NMI(\pi^i, \pi^f)$ measures the normalized mutual information between a base clustering and the final clustering π^f.

We apply the four consensus functions to each given ensemble and among the four clustering outputs we select the clustering that maximizes the SNMI value. We then compute the NMI value between the selected clustering and the class labels as the evaluation of the ensemble. In Figure 4, we plot the new performance measure of the three ensemble constructors.

Figure 4 shows that, among the three ensemble constructors examined here, the ensembles generated by RP perform best in most cases. This is consistent with what we observe in the upper-bound performance analysis. It should be noted that SNMI sometimes fails to select the optimal clustering, which can be inferred from the fact that the performance based on SNMI is in some cases significantly worse than the upper-bound performance presented in Section 6.1. Note that, among the eight data sets used in the experiments, the difference between the upper-bound performance and the SNMI-based performance is least significant for the ISOLET6 data set, for which we see the highest base-clustering quality. This suggests that SNMI may be best suited as an objective function for cluster ensembles when the base clustering quality is high.

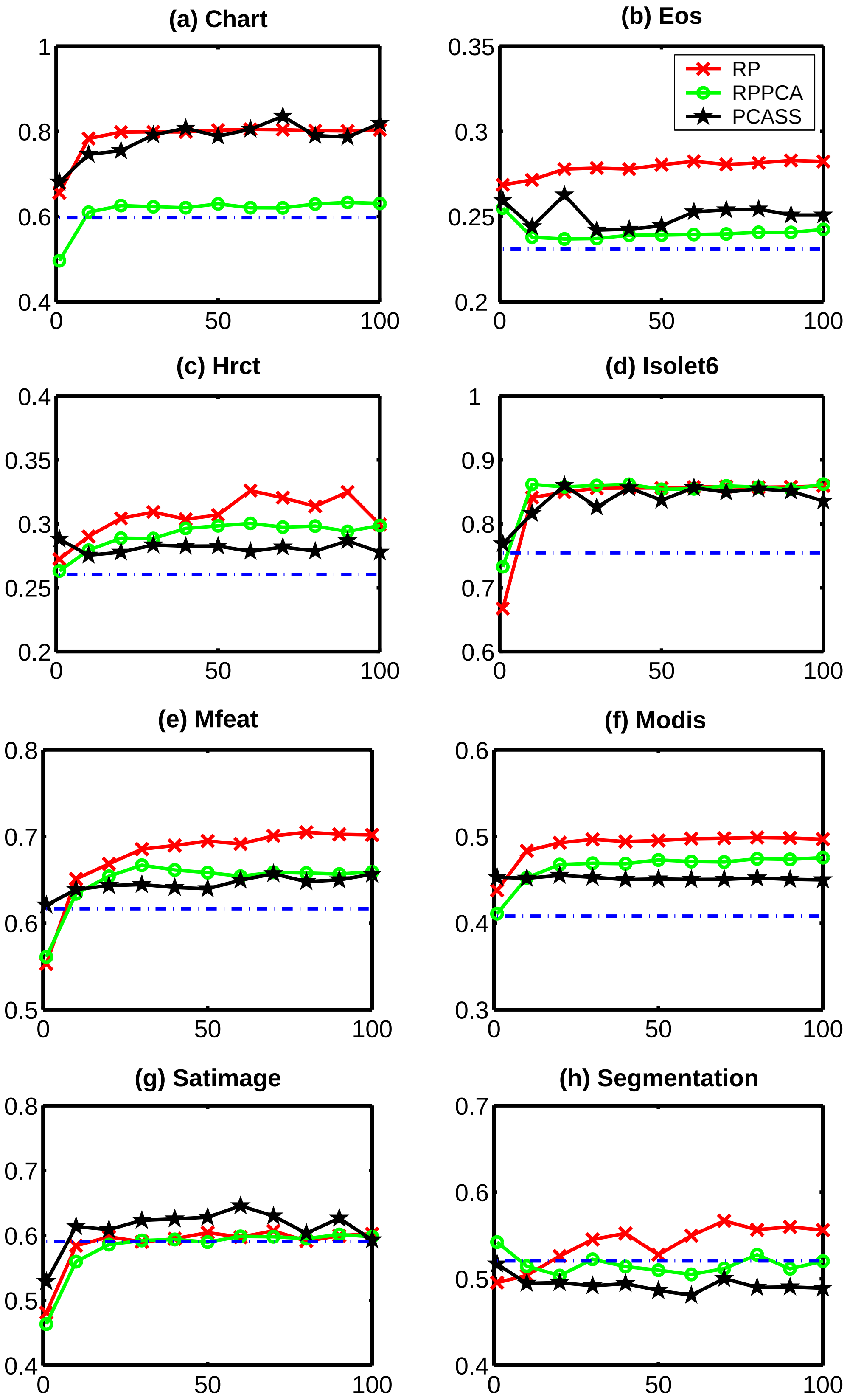


Figure 4: SNMI-based evaluation of the three types of ensembles.

7. Experiments on Consensus Functions

In this section, we evaluate the four consensus functions described in Section 4. Although we have identified RP as the most effective ensemble construction approach among the three constructors examined, we choose to evaluate the consensus functions using both RP ensembles and PCASS ensembles. This is because RP and PCASS behave differently with respect to the diversity and the quality of the resulting base clusterings. Particularly, RP produces base clusterings that are highly diverse, whereas the base clusterings of PCASS are often better in quality compared to RP. We are interested in evaluating the consensus functions using both types of ensembles and finding out if their performance is sensitive to these basic properties of cluster ensembles.

7.1 Evaluation of Consensus Functions Using RP Ensembles

To generate RP ensembles, we use the same set of parameters as described in Section 5.2. For each data set, we use RP to generate ensembles of ten different ensemble sizes. They are 10, 20, $\cdots$, up to 100 respectively. For each ensemble size, ten different ensembles are generated. This creates a pool of 100 RP ensembles for each data set. For each ensemble Π_i in the pool, we measure the quality of its base clusterings using NMI and the class labels and average them to obtain the average base-clustering quality (represented as q_i) of ensemble Π_i. To evaluate a consensus function using Π_i, we apply the consensus function to Π_i to produce a final clustering, which is then evaluated using NMI and the class labels. Denoting the obtained NMI value as f_{nmi}, we then calculate $r_i = \frac{f_{nmi}}{q_i} - 1$, which measures the rate that ensemble Π_i improves over its base clusterings when the particular consensus function was used to produce the final clustering. We use r_i as the final evaluation of the consensus function under ensemble Π_i.

For each data set, a pool of 100 RP ensembles are used to evaluate each of the four consensus functions. We evaluate the improvement rate for every ensemble in the pool and average the 100 results to obtain a final evaluation. Note that we choose not to separately present the evaluation results for different ensemble sizes to facilitate a clear presentation and interpretation of the results. More importantly, we believe that this choice will not influence our evaluation results because from Section 6 we see that the ensemble performance appears to be stable for different ensemble sizes.

Columns 2-5 of Table 5 show the improvement rates of the four consensus functions. We show the results of each data set as well as the average of all data sets. The numbers shown for each individual data set are obtained by averaging 100 results whereas the numbers shown for the overall average are obtained by average 800 results.

Table 5 also present the ranking of the four consensus functions for each data set and for the final average. Note that the notation A $<$ B indicates that the performance of A on average is inferior to B and the difference is statistically significant (according to paired *t*-test with significance level 0.05). In contrast, A $\leq$ B indicates that the average performance of A is better than that of B but the difference is not statistically significant. Finally, we highlight in boldface the best performance obtained for each row in the table.

The first observation we make is that no single consensus function is the best for all data sets. Indeed, each of the four consensus functions examined wins the top position for some data set. Among the four consensus functions, HBGF and IBGF appear to be more

Table 5: The rates of improvement over base clusterings achieved by the four consensus functions when applied to RP ensembles.

	IBGF(I)	CBGF(C)	HBGF(H)	KMCF(K)	Ranking
CHART	.225	**.248**	.232	.235	$I < H \leq K < C$
EOS	.032	-.082	**.050**	**.041**	$C < I < K \leq H$
HRCT	.043	-.063	.039	**.192**	$C < H \leq I < K$
ISOLET6	**.275**	.262	**.276**	.143	$K < C < I \leq H$
MFEAT	**.258**	.202	**.260**	.204	$C \leq K < I \leq H$
MODIS	**.119**	**.114**	**.118**	.102	$K < C \leq H \leq I$
SATIMAGE	**.276**	.179	**.260**	.171	$K \leq C < H \leq I$
SEGMENTATION	.077	**.125**	**.131**	.023	$K < I < C \leq H$
Average	.163	.123	**.171**	.139	$C < K < I < H$

Table 6: The rates of improvement over base clusterings achieved by the four consensus functions when applied to PCASS ensembles.

	IBGF(I)	CBGF(C)	HBGF(H)	KMCF(K)	Ranking
CHART	.032	.086	.055	**.164**	$I < H < C < K$
EOS	-.100	-.404	-.105	-.014	$C < H < I < K$
HRCT	-.016	-.110	**.008**	-.131	$K \leq C < I < H$
ISOLET6	.089	.083	**.099**	-.134	$K < C \leq I < H$
MFEAT	**.033**	.018	**.032**	-.048	$K < C < H \leq I$
MODIS	**.000**	-.016	-.005	-.076	$K < C < H < I$
SATIMAGE	.215	.171	**.221**	-.191	$K < C < I < H$
SEGMENTATION	-.048	**.052**	.043	-.133	$K < I < H < C$
Average	.026	-.015	**.044**	-.070	$K < C < I < H$

stable than the other two functions because on average they achieve better performance and win the top positions more frequently than the other two approaches. The performance of HBGF and IBGF is similar but on average HBGF achieves a higher improve rate of 17.1% (averaged across the eight data sets used in our experiments).

7.2 Evaluation of Consensus Functions Using PCASS Ensembles

We generate PCASS ensembles to evaluate the consensus functions using the same procedure as described in the previous section. The values of the parameters are set as described in Section 5.2.

From the results shown in Table 6, we observe that HBGF again achieves the most stable performance among the four consensus functions. On average, its improvement rate over base clusterings is 4.4%. This is consistent with the results we observe for RP ensembles. In comparing Table 5 and 6, we observe that all consensus functions see certain degradation in the improvement rates over the base clusterings for PCASS ensembles—suggesting that the consensus functions are influenced by the base-clusterings diversity and quality.

Among the four consensus functions, we note that KMCF sees the most significant degradation and fails to improve over the base clusterings for all but the CHART data set. We argue that the failure of KMCF may be directly attributed to the reduced diversity of the PCASS ensembles. In comparing the performance of KMCF across different data sets, we notice that low diversity in base clusterings tends to correspond to poor performance of KMCF. For example, among the eight data sets, KMCF sees the worst performance for the SATIMAGE and ISOLET6 data sets, which are the data sets that have the least diversity in the ensembles.

In contrast to KMCF, we see a different trend for CBGF, where poor final performance often correlates with low quality in base clusterings. Indeed, among the eight data sets, CBGF performs the worst for EOS and HRCT, and we observe that they are the data sets for which PCASS ensembles have the lowest base-clustering quality.

Finally, for HBGF and IBGF, we do not see either the quality or diversity dominating the final performance. Instead, the final performance of these two approaches appears to be influenced jointly by both factors.

In conclusion, our experiments suggest that, among the four consensus functions, HBGF and IBGF often perform better than CBGF and KMCF. The performance of HBGF and IBGF tends to be comparable, but on average HBGF performs slightly better. We also observe that KMCF is sensitive to the diversity of the ensembles—low diversity leads to poor KMCF performance. In contrast, CBGF is more influenced by the quality of the base clusterings.

8. Related Work On Cluster Ensembles

The framework of cluster ensembles was recently formalized by Strehl and Ghosh (2002). There have been many instances of the cluster ensemble framework in the literature. They differ in how they construct the ensembles (different ensemble constructor) and how they combine the solutions (different consensus function).

Examples of ensemble constructors include using different bootstrap samples (Fridlyand and Dudoit, 2001, Strehl and Ghosh, 2002), using different initializations to randomized clustering algorithms such as K-means (Fred and Jain, 2002), using different feature subsets to perform clustering (Kargupta et al., 1999, Strehl and Ghosh, 2002) and using multiple different clustering algorithms (Strehl and Ghosh, 2002). In this paper, we focused on construction approaches that build on dimension reduction techniques in attempt to address problems caused by high dimensionality. We identified RP as an effective ensemble constructor. Further, we conducted an in-depth analysis of the properties of the ensembles generated by different constructors, providing an explanation for the effectiveness of the RP ensemble constructor.

A main focus of the cluster ensemble research has been placed on combining clusterings and a variety of consensus functions are available in the literature. In this paper we do not intend to cover all existing consensus functions in our comparisons. Instead, we chose to examine four consensus functions that have been shown to be highly competitive compared to other alternatives (Strehl and Ghosh, 2002, Topchy et al., 2004, Fern and Brodley, 2004). Below we review various alternative consensus functions and relate them to those examined in this paper.

Many existing techniques for combining clusterings bear strong similarity with the IBGF approach in the sense that they generate an instance similarity matrix based on a given cluster ensemble. The difference lies in how the matrix is used to produce the final clustering, for which various techniques, such as simple thresholding (Fred, 2001) and agglomerative clustering algorithms (Fred and Jain, 2002, Fern and Brodley, 2003, Monti et al., 2003), have been examined.

There also exist a number of cluster-based approaches that are similar in essence to CBGF. These approaches seek to find a correspondence structure among clusters that are formed in the ensemble and use a voting scheme to assign instances to clusters once a correspondence structure is found. CBGF finds such a correspondence structure by partitioning a graph describing the similarity relationship among clusters. Other approaches include using a relabeling process (Dudoit and Fridlyand, 2003) to map the clusters to virtual class labels, and clustering the cluster centers to group similar clusters together (Dimitriadou et al., 2001).

9. Conclusions and Future Work

In this paper, we studied cluster ensembles for high dimensional data clustering. First, we examined three different ensemble constructors for generating cluster ensembles, which build on two popular dimension reduction techniques: random projection (RP) and principal component analysis (PCA). We empirically evaluated the ensemble constructors on eight data sets, and concluded that cluster ensembles constructed by RP perform better for high dimensional clustering in comparison to 1) the traditional approach of a single clustering with PCA dimension reduction and 2) ensembles generated by the other two ensemble constructors examined in the paper. In an attempt to explain the superiority of RP ensembles, we analyzed the diversity and quality of the ensembles generated by different constructors. Our analysis indicated that if the base clusterings have similar quality, the higher the diversity, the better the ensemble can potentially perform. Further, high diversity may also compensate for low base clustering quality. These results provide strong evidence that RP's superior performance can be attributed to the fact that it produces highly diverse base clusterings.

We also examined four different consensus functions for combining clusterings. We used two different types of cluster ensembles (RP and PCASS ensembles) to evaluate the four consensus functions. We identified two of the graph partitioning based approaches, the HBGF and IBGF approaches, as superior to the other two approaches. On average, HBGF improved over base clusterings by 17.1% for RP ensembles and by 4.4% for PCASS ensembles, whereas IBGF improved by 16.3% and 2.6% respectively. Overall, HBGF slightly outperformed IBGF. The experiments suggested that all of the four consensus functions are influenced by the quality and diversity of the base clusterings. We saw evidence that KMCF is highly sensitive to the diversity of the base clusterings. Therefore, if a cluster ensemble lacks diversity, KMCF should be avoided as the choice for consensus function. In contrast, CBGF is sensitive to the quality of the base clusterings. And finally, the quality and diversity of base clusterings jointly impact HBGF and IBGF, making them more stable when facing the trade-off of the quality and diversity of the base clusterings.

To conclude, we have identified random projection as an effective solution for generating cluster ensembles and HBGF as a good candidate for combining the clusterings. For future research, we note that, in this paper as well as previous research on cluster ensembles, all base clusterings are treated equally by consensus functions. There is no doubt that some of the base clusterings may be better than others. An interesting question is whether we can improve the consensus functions by assigning weights to the base clusterings according to their quality. Finally, we note that the empirical performance of cluster ensembles tends to level off as the ensemble size increases. This indicates that very large ensemble sizes may be unnecessary. However, it remains an open question how can we decide a proper ensemble size when applying cluster ensembles in practice.

References

R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, pages 94–105. ACM Press, 1998.

F. R. Bach and M. I. Jordan. Learning spectral clustering. In *Advances in Neural Information Processing Systems 16*, 2003.

R. Bellman. *Adaptive control processes.* Princeton University Press, 1961.

K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is nearest neighbors meaningful? In *Proceedings of the International Conference on Database Theories*, pages 217–235, 1999.

C. L. Blake and C. J. Merz. UCI repository of machine learning databases, 1998. URL `http://www.ics.uci.edu/∼mlearn/MLRepository.html`.

T. M. Cover and J. A. Thomas. *Elements of Information Theory.* John Wiley & Sons, 1991.

S. Dasgupta. Experiments with random projection. In *Uncertainty in Artificial Intelligence: Proceedings of the Sixteenth Conference (UAI-2000)*, pages 143–151. Morgan Kaufmann, 2000.

I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Knowledge Discovery and Data Mining*, pages 269–274, 2001.

T. G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting and randomization. *Machine learning*, 2:139–157, 2000.

E. Dimitriadou, A. Weingessel, and K. Hornik. Voting-merging: An ensemble method for clustering. In *Proceedings of International Confernce on Artificial Neural Networks*, pages 217–224, 2001.

S. Dudoit and J. Fridlyand. Bagging to improve the accuracy of a clustering procedure. *Bioinformatics*, 19, 2003.

J. G. Dy and C. E. Brodley. Feature selection for unsupervised learning. *Journal of Machine Learning Research*, 5:845–889, 2004.

J. G. Dy, C. E. Brodley, A. Kak, C. Shyu, and L. S. Broderick. The customized-queries approach to CBIR using EM. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 400–406. IEEE Computer Society Press, 1999.

X. Z. Fern and C. E. Brodley. Random projection for high dimensional data clustering: A cluster ensemble approach. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 186–193, 2003.

X. Z. Fern and C. E. Brodley. Solving cluster ensemble problems by bipartite graph partitioning. In *Proceedings of the Twenty First International Conference on Machine Learning*, pages 281–288, 2004.

A. L. N. Fred and A. K. Jain. Data clustering using evidence accumulation. In *Proceedings of International Conference on Pattern Recognition*, 2002.

A.L.N. Fred. Finding consistent clusters in data partitions. In *Proceedings of the Third International Workshop on Multiple Classifier Systems*, pages 309–318, 2001.

J. Fridlyand and S. Dudoit. Applications of resampling methods to estimate the number of clusters and to improve the accuracy of a clustering method. Technical Report 600, Statistics Department, UC Berkeley, 2001.

M. Friedl, D. McIver, J. Hodges, X. Zhang, D. Muchoney, A. Strahler, C. Woodcock, S. Gopal, A. Schneider, A. Cooper, A. Baccini, F. Gao, and C. Schaaf. Global land cover mapping from modis: algorithms and early results. *Remote Sensing of Environment*, 83: 287–302, 2002.

J. H. Friedman. Exploratory projection pursuit. *Journal of the American Statistical Association*, 82(397):249–266, 1987.

K. Fukunaga. *Statistical Pattern Recognition (second edition)*. Academic Press, 1990.

R. Hecht-Nielsen. Context vectors: general purpose approximate meaning representations self-organized from raw data. In *Computational Intelligence: Imitating Life*, pages 43–56. IEEE press, 1994.

S. Hettich and S. D. Bay. The UCI KDD archive, 1999. URL `http://kdd.ics.uci.edu`.

P. J. Huber. Projection pursuit. *Annals of Statistics*, 13(2):435–475, 1985.

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

S. Monti, P. Tamayo, J. Mesirov, and T. Golub. Consensus clustering: A resampling-based method for class discovery and visualization of gene expression microarray data. *Machine Learning*, 52:91–118, 2003.

A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14*, pages 849–856, 2002.

J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

A. Strehl and J. Ghosh. Cluster ensembles - a knowledge reuse framework for combining multiple partitions. *Machine Learning Research*, 3:583–417, 2002.

A. Topchy, A. K. Jain, and W. Punch. Combining multiple weak clusterings. In *Proceedings IEEE International Conference on Data Mining*, pages 331–338, 2003.

A. Topchy, A. K. Jain, and W. Punch. A mixture model for clustering ensembles. In *Proceedings of SIAM Conference on Data Mining*, pages 379–390, 2004.