WEIGHTED CLUSTERING ENSEMBLES

by

Muna Saleh Al-Razgan
A Dissertation
Submitted to the
Graduate Faculty
of
George Mason University
In Partial fulfillment of
The Requirements for the Degree
of
Doctor of Philosophy
Information Technology

Committee:

_____	Dr. Carlotta Domeniconi, Director
_____	Dr. Daniel Barbará, Committee Member
_____	Dr. Jessica Lin, Committee Member
_____	Dr. Harry Wechsler, Committee Member
_____	Dr. Daniel Menasce, Associate Dean for Research and Graduate Studies
_____	Dr. Lloyd J. Griffiths, Dean, School of Information Technology and Engineering
Date: May 15, 2008	Summer Semester 2008 George Mason University Fairfax, VA

Weighted Clustering Ensembles

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University

By

Muna Saleh Al-Razgan
Master of Science
George Mason University,
Bachelor of Science
College of Education for Girls, Saudi Arabia,

Director: Dr. Carlotta Domeniconi, Professor
Department of Computer Science

Summer Semester 2008
George Mason University
Fairfax, VA

Dedication

I dedicate this dissertation to my dearest parents (Nora, Saleh), and to my grandparents (Hiela, Ali) for their support and prayer in achieving this goal. I am grateful that they were there for me, and knowing they are proud of my accomplishment is a great honor.

I would also like to dedicate this dissertation to my lovely sister Manar and to my dear bother Abdullah.

Acknowledgments

My greatest thanks and gratitude is to My LORD Allah, the most beneficial and merciful for His blessing during all the difficulties to achieve my goal and accomplishing this task.

I offer my deep acknowledgment to my advisor Dr. Carlotta Domeniconi, who believed in me, guided me through the writing process, and enlightened my thinking.

I would like to thank Dr. Daniel Barbará, Dr. Jessica Lin, and Dr. Harry Wechsler for serving in my committee.

My special thanks go to my friend Reba Elliott for her dedication in editing my paper from the beginning of the proposal phase to the end. Her commitment was extremely significant in this dissertation. Her friendship will never be forgotten.

My special appreciation goes to my friends, classmates, and colleagues: Dr. Aynur Abdurazik, Muhammad Abdulla, Dr. Marym Al-Ali, Fawzia Al-Rashid, Dr. Nazik Zakari, and Dr. Amani Babgi. They continued to support me, cheered me during difficult time, and ultimately were with me when I need them the most. I will never forget their commitment, friendship, and support.

I would also like to thank many friends I got to know while studying my PhD at George Mason University: Loulwah AlSumait, Malak Alnory, Erika Olimpiew, Dalal Alarayed, Ghada Alnifie, Shen-Shyang Ho, and Bojun Yan, to name a few.

Last but not least, very special thanks go to my family. My father who encouraged me to reach my dreams, my mother whose prayer and words of wisdom helped through life, to my grandparents, for their continuous prayers and loving during this process. To my loving brothers and sisters: Nawal, Othman, Abdullah, and Manar, whose endless support and encouragements made my work an easier task to complete. To my aunt Latifa, for her continuous phone calls encouraging me and believing in me. Finally, to my aunt Um-Mohmmad Naimah who surrounded me with words of wisdom and reminding me of the saying "if there is a will, there is a way" to accomplish this work.

Table of Contents

List of Tables

List of Figures

Abstract

WEIGHTED CLUSTERING ENSEMBLES

Muna Saleh Al-Razgan, PhD

George Mason University, 2008

Dissertation Director: Dr. Carlotta Domeniconi

Clustering is a popular approach to exploratory data analysis and mining. However, clustering faces difficult challenges due to its ill-posed nature. First, it is well known that off-the-shelf clustering methods may discover different patterns in a given set of data, because each clustering algorithm has its own bias resulting from the optimization of different criteria. Second, there is no ground truth against which the clustering result can be validated. High dimensional data also pose a difficult challenge to the clustering process. Various clustering algorithms can handle data with low dimensionality, but as the dimensionality of the data increases, these algorithms tend to break down. In this dissertation, we introduce novel clustering ensemble techniques and novel semi-supervised approaches to address these problems.

Clustering ensembles offer a solution to challenges inherent to clustering arising from its ill-posed nature: they can provide more robust and stable solutions by making use of the consensus across multiple clustering results, and they can average out the emergent spurious structures which arise due to the various biases of each participating algorithm, and due to the variance induced by different data samples. We introduce and analyze three new consensus functions for ensembles of subspace clusterings. The ultimate goal of our consensus functions is to provide hard partitions of the data, and weight vectors which convey information regarding the subspaces within which the individual clusters exist. We demonstrate the effectiveness of our three techniques by running experiments with several real datasets, including high dimensional text data, and investigate the issue of diversity and accuracy in our ensemble techniques.

We also study scenarios in which limited knowledge on the data (in terms of pairwise constraints) is available from the user. We develop a methodology to embed such constraints into the ensemble components, so that the desired structure emerges via the consensus clustering. We introduce a mechanism which leverages the ensemble framework to bootstrap informative constraints directly from the data and from the various clusterings, without intervention from the user. We demonstrate the effectiveness of our proposed techniques with experiments using real datasets and other state-of-the-art semi-supervised techniques.

Chapter 1: Introduction

1.1 Data Mining Overview

Advances made in data collection and improvements achieved in database technology have contributed to the amassing of data in recent years. These data include Web data, e-commerce records, tallies of purchases at department and grocery stores, and credit card transactions. However, the amassing of data has not led to a corresponding increasing of knowledge. With such a large amount of data, the extraction of useful information is a difficult task. To solve this problem, data mining strategies have been developed that automatically extract interesting knowledge from datasets [33].

Data mining extracts "implicit, previously unknown and potentially useful information" from data [62]. Thus, data mining contributes to knowledge discovery, the process of translating the raw data in a database into useful knowledge. Data mining is a multi-disciplinary field that draws from areas such as statistics, data warehousing, machine learning, information retrieval, and pattern recognition. Data mining helps us answering questions that database queries cannot. For example, retailers could identify what type of customers respond to mass marketing, or financial institutions could model and predict market behavior.

1.2 Data Mining Techniques

To meet the difficulties and challenges of dealing with large amounts of data with high dimensionality, consistently accurate techniques must be developed. These techniques must be efficient and scalable to handle large data sets with different formats. Research inquiring into the development of such techniques has been done. Some of these techniques were borrowed from advances made in the fields of statistics, pattern recognition, and machine learning. Data mining owes a particular debit to research in sampling, search algorithms, query processing, and modeling techniques. Several different data mining techniques for classification, association rules, and clustering have been developed. Data mining tasks are usually grouped into two types, predictive (classification) and descriptive (association rule discovery, clustering).

Classification is the task of building a set of models that identify classes or concepts. These models are then used to predict the class labels of unknown or new objects [33]. To establish a classification model, one trains the model using n training data, with their J classes. The training observations consist of D attribute measurements $\mathbf{x} = (x_1, ..., x_D) \in \Re^D$ and the known class labels $\mathbf{x} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ where $y_i \in \{1, ..., J\}$. Training data are samples drawn from the underlying unknown distribution. Test data are also vector of observations, but without predefined class labels. The objective of the classification model is to predict the class label of newly coming data points. Classification models have been utilized in many disparate fields such as marketing, medical analysis, and information retrieval [62].

Association rules discover patterns of dependence, by predicting the likelihood of the occurrence of a given item based on the occurrence of other items. Given a set of items of inventory $Q = \{t_1, ..., t_m\}$ that contains all items in a market basket data and a set of all transactions $D = \{d_1, ..., d_n\}$, each transaction d_i contains a set of

items of inventory from Q such that $d_i \subseteq Q$. The objective of association rules is to find rules that correlate the attendance of one set of items with the attendance of another set of items. For example, a transaction d contains a set of items X. An association rule can be represented as $X \Rightarrow Y$, such that $X, Y \subseteq Q$, and $X \cap Y = \emptyset$. The rule $X \Rightarrow Y$ holds with support s, where s indicates the probability of the item sets X and Y in the data set, $P(X \cup Y)$. Support s is the number of transactions that contains both X and Y divided by the number of transactions in the dataset. The rule $X \Rightarrow Y$ has confidence c, where c is probability that Y occurs in transactions already containing X, $P(Y|X)$. Confidence c is the number of times X and Y occur together in the transactions divided by the number of transactions containing X.

This method is widely used in market basket, or transaction data analysis. For example, association rules might discover that customers who buy bagels are likely to also buy cheese, or might discover that other items are frequently purchased together by customers. This kind of analysis requires a large amount of several computations as combinations of the entire dataset need to be analyzed to find different patterns of interest [33].

Sometimes training data are not available in advance, or one has a huge amount of data without any related ground truth. In order to extract useful information from these kinds of datasets, a technique that works in an unsupervised fashion is needed. One of these techniques is clustering. Clustering groups data points which are similar to each other in one cluster, and places data points which are not similar to each other in different clusters. The process works according to similarity measures, and different similarity measures partition the data in different structures. A few examples of similarity (distance) measures are the Euclidean distance [62], the Pearson correlation [62], and the Jaccard index [62]. The choice of a specific similarity measure is particularly important, as it drives the structure being discovered

in the data. Most clustering algorithms find different structures in a given dataset based on different optimization criteria. Clustering is a studied problem in disciplines such as pattern recognition, image processing, marketing, and statistics. Wide usage of clustering algorithms proves their usefulness in exploratory data analysis [40].

In this study we will look at clustering methods in more detail.

1.3 Challenges

Although clustering has proven useful in several fields, significant challenges remain to be met. Most clustering techniques do not produce acceptable results due to issues with clustering algorithms and data distributions.

Issues with clustering algorithms

There are two significant challenges inherent to clustering algorithms. First, various clustering algorithms find different structures (e.g., size, shape) in the same dataset. This is because each individual clustering algorithm has its own preferences due to the optimization of different criteria. Second, a single algorithm with different parameter settings can find various structures on the same dataset. Since no labeled data are available, no cross-validation can be used to tune the parameters. These challenges confront the user, making the selection of a proper clustering technique very difficult. He or she is left without any guidance for selecting the appropriate clustering technique or parameter values for a given set of data.

Issues with data distributions

- There are usually many samples to be clustered, and with a large amount of data, scaling issues must be considered. Therefore, an algorithm that scales linearly with the number of samples is desirable.

- Some datasets have a significant number of outliers. Failure to detect these

outliers adversely affects the clustering result, and leads to the discovery of inaccurate structures.

An additional difficulty is introduced when the data belong to a high dimensional space. Often clusters of data are embedded in subspaces comprised of a subset of original feature set. Different cluster may exist in different subspaces, in which finding regions populated with a large number of points becomes difficult. Researchers developed dimensionality reduction techniques in order to reduce the dimensionality of the feature space. However, well known dimensionality reduction techniques are global, and thus unable to capture local structure in the data. In order to overcome this problem subspace clustering algorithms have been introduced. Subspace clustering algorithms locate clusters in subspaces of the original space. Applying subspace clustering techniques avoids the loss of local information inherent to global dimensionality reduction techniques. However, the overwhelming majority of these methods depend on input parameters such as the number of dimensions per cluster, density thresholds, etc. Often, these parameters are unknown in advance to the user.

A solution to the challenges of clustering and subspace clustering methods remains an ultimate goal. As part of the endeavor to reach this goal, researchers devised the process of combining different clusterings into a single clustering, a process known as "cluster ensembles." Cluster ensembles can provide more robust and stable solutions across different domains and datasets. However, designing a proper consensus function for cluster ensembles is far from trivial. The design of an effective consensus function is a difficult task since data are unlabeled, and thus there is no well defined correspondence between the different clusterings of a given dataset.

Another avenue to address the ill-posed problem of clustering is through the use of semi-supervised clustering where some information about the data is indeed available. Semi supervised clustering uses prior knowledge to guide the clustering process, and

to provide results that adhere to the user's preference. In this way, semi-supervised clustering techniques promise an ease of use and a natural approach along with accurate results. However, there are some open challenges to semi supervised clustering. These challenges are: identify reliable and useful constraints under limited resources (e.g limited access to an oracle). Labels for the dataset are generally unavailable or prohibitively expensive to obtain. However, an end user might be able to provide pairs of similar and dissimilar examples. For instance, a human expert may be able to identify two text documents that discuss similar or related topics. The question then becomes how to leverage this minimal knowledge to accurately group the remaining members of the dataset. This situation, which is known as semi-supervised clustering, has attracted researchers.

The challenges of combining partitions and producing better overall clustering results with or without prior knowledge are under intense study by researchers. The discovery of new approaches to clustering is the focus of our research.

1.4 Contributions

The goal of this dissertation is to improve upon cluster analysis. We overcome some of the challenges inherent to clustering and subspace clustering by designing new approaches for clustering ensembles and semi-supervised clustering. A summary of our contributions follows:

- *Weighted Similarity Partitioning Algorithm* (WSPA): Defines a consensus function for an ensemble of clusterings by mapping the problem onto a k-way partitioning problem. This method measures pairwise similarities in different subspaces of the input space. No assumption is made about the underlying distribution of the data, thereby avoiding parameter estimations. It is shown that

this ensemble technique can produce partitions that are as good as or better than the best individual clustering.

- *Weighted Bipartite Partitioning Algorithm* (WBPA): Maps the problem of finding a consensus partition to a bipartite graph partitioning problem with weight values ranging to [0,1]. This method has a conceptual advantage in that partitions both cluster vertices and instance vertices simultaneously. To the best of our knowledge, WBPA and WSPA are the first attempts to design cluster ensembles for subspace clustering. It is shown that this ensemble technique can produce partitions that are as good as or better than the best individual clustering.

- *Weighted Subspace Bipartite Partitioning Algorithm* (WSBPA): This method is an extension of WBPA: it provides hard partitions of the data along with weight vectors that convey information regarding the subspaces within which the individual clusters exist. To the best of our knowledge, this technique is the first attempt in the literature to produce subspace clustering results within the context of ensemble research. The weight vector produced by WSBPA captures the local relevance of features within each cluster.

- *Categorical Similarity Partitioning Algorithm* (CSPA) and *Categorical Bipartite Partitioning Algorithm* (CBPA): These methods are generalizations of our WSPA and WBPA techniques to be used with categorical data. These methods compute the distance between a point and a cluster in categorical data by considering the Jaccard distance. It is shown that these ensemble methods filter spurious structures, and achieve better results with respect to individual base clusterings.

- *Constrained Locally Adaptive Clustering* (CLAC): Embeds constraints into the

initialization and iterative phases of subspace clustering. This algorithm represents an attempt to incorporate constraints, which capitalize on information known to a human expert, to mitigate the ill-posed nature of clustering and allow an end user to tune preferences. It is shown that this algorithm, because it is capable of handling high-dimensional data, is an improvement over other semi-supervised clustering algorithms.

- *Constrained-Weighted Bipartite Partitioning Algorithm* (C-WBPA): Embeds knowledge-based constraints during the partitioning process of each ensemble component to improve the quality of the overall ensemble. By enforcing the resulting constraints at the components level, it ensures that the corresponding structure they represent is carried into the consensus function, and therefore into the consensus partition. C-WBPA produces a robust and stable solution of the given data that adheres to the users preference.

- *Bootstrapping of Constraints: Penta-training* Bootstraps constraints from the data for subspace clustering ensembles where few or no constraints are provided by the user. This method makes use of the collaborative knowledge produced by ensemble to generate constraints directly from the data. To best of our knowledge, this is the first attempt to bootstrap constraints using clustering ensembles.

  The efficacy of the techniques presented is demonstrated through experimental evaluations, using a variety of simulated and real world problems. In addition, we investigate in great detail the issue of diversity and accuracy for our ensemble techniques. Our results reveal that high diversity signifies high accuracy.

Chapter 2: Clustering Algorithms

2.1 Introduction

Clustering is the process of discovering homogeneous groups or clusters according to a given similarity measure. Clustering maximizes intra-connectivity among patterns in the same cluster while minimizing inter-connectivity between patterns in different clusters. Connectivity is measured using the attribute values that represent the objects in the dataset [46].

Definition: Given a set of n patterns in D-dimensional space, $\mathbf{x}_i = (x_1, ..., x_D) \in \Re^D$, $i = 1, ..., n$, Clustering algorithms find a partition of the data into k clusters that achieves a required objective, defined in terms of a given similarity (distance) measure $d(\mathbf{x}_i, \mathbf{x}_j)$. The clustering algorithm partitions the data such that patterns in one cluster are more similar to each other than to patterns in different clusters.

Because the measurement of connectivity and the succeeding grouping are performed without knowledge about the patterns' class labels, clustering is known as an unsupervised method. This unsupervised quality is one of several challenges facing the researcher. One must also select an algorithm, a similarity measure, criterion function, and initial condition, all of which must provide clusters of suitable size, shape, and density [46].

Clustering is an important technique in discovering meaningful groups of data points. Clustering provides speed and reliability in grouping similar objects in very large datasets, which makes it a remarkably effective method for use in multiple applications. The exploratory character of clustering makes it well suited to data mining

techniques such as database segmentation, predictive modeling, and visualization [41]. Both because it provides significant benefit to the study of data structures, and because advances remain to be made in the field of clustering study, clustering is an ideal candidate for research [40].

There exist several clustering algorithms. The two most common methodologies to perform clustering are hierarchical and partitioning. Hierarchical clustering algorithms group data objects by similarity. They create sets of nested clusters, producing a hierarchical tree [33]. Unlike the vertical organization of hierarchical clustering, partitional clustering organizes data into sets based on data density. These sets are selected by the algorithm to ensure minimal within-cluster scatter and maximal between-cluster scatter [40].

2.1.1 Hierarchical Clustering Algorithms

Hierarchical clustering algorithms produce trees, called dendograms, that represent nested groups of data organized hierarchically. This algorithm does not require information about the number of clusters, but does require information about where to cut the dendogram to produce final clustering results. Hierarchical clustering algorithms can be classified as either agglomerative or divisive. The agglomerative technique begins by considering each object to be a cluster, and then gradually merges similar clusters until the termination condition is met. The divisive technique, however, begins with all patterns grouped into one cluster, and then performs a splitting process until the termination condition is achieved [41]. No matter which technique is used, the resulting dendogram reveals cluster-subcluster relationships, and the order in which the clusters were merged or split [62].

Agglomerative hierarchical clustering can be approached by three methods: single-link, complete-link, and average-link. The single-link method considers the distances

between two clusters based on the closest points in those clusters. The complete-link method considers the most distant points between two clusters. The average-link method considers the average pairwise distance between all points in two clusters.

While hierarchical clustering does have few strengths (it does not require information about the number of clusters, for example) it also possesses significant drawbacks. First, the computation necessitated by hierarchical clustering is $O(n^3)$, and the storage required is $O(n^2)$ where n is the number of data points, since a distance matrix must be calculated at each step. Second, decisions about merging or splitting clusters cannot be reversed. Third, hierarchical clustering lacks a global objective function. These drawbacks pose concerns for the researcher [62].

2.1.2 Partitioning Clustering Algorithms

Partitioning clustering algorithms partition the dataset into clusters such that points in one cluster are more similar to each other than to points in different clusters.

Partitioning clustering divides a dataset into k number of clusters. The partitioning must satisfy two conditions: each cluster must contain at least one point, and each point must belong to only one cluster. Searching the entire dataset for all possible partitions, a search necessary for global optimization, is prohibitively expensive. A local optimization method is called for, the most commonly used of which is k-means [33].

k-means clustering is relatively simple. First, it selects randomly k instance as centroids. Second, it assigns the remaining instances to the closest centroids. Third, it updates the centroids for each cluster by taking the mean of all points in that cluster. Finally, it iterates until no changes are made to the centroids (there is no reassignment of any point from one cluster to another) [46].

k-means clustering, like all partitioning, avoids many of the pitfalls inherent in

hierarchical clustering, and so is used more frequently by researchers. Partitioning methods optimize objective functions. k-means, for example, minimizes the sum of the squared distances between all points in a cluster and their centroids to find compact and dense groups. Partitioning methods decisions are reversible, and the intensity of the calculation is less than in hierarchical, requiring a much more modest amount of storage space and time. k-means stores only the data points and centroids, therefore the storage required is $O((n+k)D)$, where n is the number of data points, k is the number of clusters, and D is the number of attributes. In addition, the time complexity is $O(IknD)$ where I is the number of iterations required to reach convergence [62].

Another partitioning algorithm is Fractal clustering [11] which discovers clusters of arbitrary shape based on the notion of fractal dimension [59]. It is an incremental technique that adds points to clusters, specified through an initial process as cells in a grid. A point is assigned to the cluster that has the minimum fractal impact, that is the cluster whose fractal dimension changes the least when the point is added. Fractal clustering provides a partition of the data, where each cluster is associated with the corresponding fractal dimension.

In general, the partitioning process is sensitive to the initial choice of centroids, and it tends to break down when the data have high dimensionality. Partitioning algorithms may produce increasingly poor results as the dimensionality increases [62].

2.2 The Curse of Dimensionality

The phrase *the curse of dimensionality* was introduced by Bellman in 1961. It refers to the increase in the sparsity of data as dimensionality increases. In high dimensional spaces, finding regions of dense points becomes a difficult task. Contrary to

data in low dimensional space, where clusters can be found easily and patterns can be easily recognized, data in space of four or more dimensions is less dense, and so not easily grouped. Part of this difficulty lies in the fact that data distributed over high dimensions become nearly equidistant, such that for any single data point the distances to its nearest and farthest neighbors are almost equal [56]. Thus, in high dimensional spaces, clustering algorithms that equally use all features are likely to be not effective. In high dimensional spaces, in fact, it is highly likely that, for any given pair of points within the same cluster, there exist at least a few dimensions on which the points are far apart from each other. In high dimensional spaces many dimensions may be irrelevant; since clustering algorithms are based primarily on similarity measures, most clustering algorithms fail in high dimensional spaces. In order to surmount the high dimensionality problem, dimensionality reduction techniques have been proposed. These methods have been successful to a certain degree. The most popular methods of feature transformation is Principal Component Analysis (PCA).

Principal Component Analysis (PCA) combines the original dimensions into fewer dimensions. The dimensions constructed by PCA are linear combinations of the original dimensions, ordered by nondecreasing variance of the data, and allow the algorithm to successfully discover latent structures in the dataset [56].

While PCA succeeds in reducing the dimensionality, it selects features globally. Thus, it fails when different groups of data clusters with respect to different subsets of dimensions. In this situation, we may not be able to filter out too many features without incurring a loss of crucial information. This is because each feature may be relevant for at least one cluster.

To illustrate the problem inherent in global dimensionality reduction techniques, we can consider a simple example of two clusters distributed according to a two dimensional Gaussian. In Figure 2.1, each cluster is closely grouped with respect to

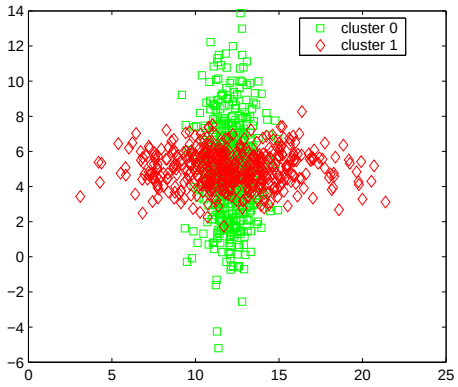

Figure 2.1: Two Gaussian Clusters

one dimension. Points in cluster 0 are closely clustered along the horizontal dimension, and points in cluster 1 are closely clustered along the vertical dimension. Therefore, pruning either one of these dimensions will result in the loss of crucial information, and will prevent the discovery of the two groups of data.

Global dimensionality reduction techniques are unable to capture local structure in the data. Thus, a proper feature selection procedure should operate locally in input space. Local feature selection allows one to embed different distance measures in different regions of the input space; such distance metrics reflect local correlations of data. Subspace clustering algorithms provide local feature selection [56].

2.3 Subspace Clustering Algorithms

Clusters may exist in multiple subspaces composed of multiple combination of dimensions. In many real-world problems, some points are correlated with respect to a given set of dimensions, while others are correlated with respect to different dimensions. Each dimension could be relevant to at least one of the clusters.

Recently, many different subspace clustering methods have been proposed [56].

They all attempt to dodge the curse of dimensionality which affects any algorithm in high dimensional spaces. Subspace clustering extends the rationale of feature selection, by locating clusters in subspaces of the original space. Because subspace clustering looks for clusters in subspaces, it is well suited for high dimensional spaces. Its use avoids the loss of information inherent to global dimensionality reduction techniques.

However, the number of subspaces is exponential in the number of dimensions. Thus, an exhaustive search of all the subspaces is not computationally feasible, and a more effective technique is necessary. A group of techniques exists that derives heuristics based on the identification of subspaces densely populated with data. There are two major types of search algorithms based on density: top-down search and bottom-up search [56]. In the following we outline the major techniques in each of these two categories.

2.3.1 Bottom-Up Search Methods

The bottom-up subspace search method is based on the presence of dense units in all $(k-1)$ dimensions when dense units are present in k dimensions. Based on this property, subspaces in two dimensions which contain dense units can be identified. Neighboring dense units are then combined to form clusters. The primary bottom-up search method is CLIQUE [3]; a closely related modification is ENCLUS [16].

CLIQUE [3] is a grid- and density-based subspace clustering method. CLIQUE first divides each dimension into the same number ξ of equal length intervals, which creates non-overlapping rectangular units. These units are considered dense if the density of the points within them exceeds a certain threshold τ. The algorithm then finds adjacent dense units and connects them to create clusters. ENCLUS [16] is closely related to CLIQUE. It is based on the idea that a subspace which contains

clusters has lower entropy than a subspace which does not.

Both CLIQUE and ENCLUS scale well with changing numbers of instances and dimensions between datasets. However, both techniques require some input parameters from the user, whose values are not easily known in advance. CLIQUE requires as input parameters the grid size and the density threshold. ENCLUS requires as input parameters the grid interval size and the entropy threshold [56].

2.3.2 Top-Down Search Methods

The top-down subspace search method begins by considering an initial estimation of clusters in all dimensions. It first assigns an equal amount of weight to each dimension for each cluster; in succeeding iterations, the algorithm assigns an updated weight to each dimension. These updated weights are used to discover the structure of each cluster. A popular approach that falls in this categories is projected clustering, which projects the data on the relevant features, and discards irrelevant features for each cluster [56].

Projected clustering considers all features in the first iteration and in the following iterations assigns a binary weight for each dimension. The dimensions with which the clusters are highly correlated receive a weight of one, and the dimensions with which the clusters are uncorrelated (or have low correlation) receive a weight of zero, and are discarded.

Projected clustering algorithms were introduced to avoid the problem of methods such as CLIQUE [3] and ENCLUS [16], which do not produce hard partitioning of the data. Applying these algorithms often results in overlapping clusters, as dense regions are projected on lower dimensions, and reported. Contrary to this, projected clustering produces hard partitions of the data, such that each subset of data points is tightly clustered in a subspace of the dimensions [1]. Projected clustering algorithms

include PROCLUS [1] and ORCLUS [2].

PROCLUS [1] seeks to select subsets of dimensions such that the points within them are densely clustered. The algorithm requires input parameters (the number of clusters and the average number of dimensions for each cluster) from the user. PROCLUS begins by choosing k number of medoids randomly from the dataset. It then applies an iterative hill climbing procedure, discarding "bad" medoids. PROCLUS searches for the dimensions most relevant to each cluster by considering those along which data are tightly clustered around the medoid's coordinate.

ORCLUS [2] is an extension of the method employed by PROCLUS. ORCLUS looks for hidden subspaces that contain clusters of inter-attribute correlations. It further extends the process of PROCLUS in that it merges clusters and selects for each cluster principal components instead of attributes [56].

In order to eliminate the need for setting the number of dimensions and avoid a possible loss of information, LAC (Locally Adaptive Clustering) [19, 20] considers all features in the input space, but properly assigns a weight value to each feature within each cluster. We describe LAC in more details in the next section.

2.3.3 Locally Adaptive Clustering (LAC)

Locally Adaptive Clustering (LAC) is a *soft* feature selection procedure that assigns weights to features according to the local variance of data along each dimension. Dimensions along which data are loosely clustered receive a small weight, which has the effect of elongating distances along that dimension. Features along which data manifest a small variance receive a large weight, which has the effect of constricting distances along that dimension. Thus the learned weights perform a directional local reshaping of distances which allows a better separation of clusters, and therefore the discovery of different patterns in different subspaces of the original input space.

Let us consider a set of n points in some space of dimensionality D. A *weighted cluster* C is a subset of data points, together with a vector of weights $\mathbf{w} = (w_1, \ldots, w_D)^t$, such that the points in C are closely clustered according to the L_2 norm distance weighted using $\mathbf{w}$. The component w_j measures the degree of correlation of points in C along feature j. The problem is how to estimate the weight vector $\mathbf{w}$ for each cluster in the dataset.

In traditional clustering, the partition of a set of points is induced by a set of *representative* vectors, also called *centroids* or *centers*. The partition induced by discovering weighted clusters is formally defined as follows.

Definition: Given a set S of n points $\mathbf{x} \in \Re^D$, a set of k centers $\{\mathbf{c}_1, \ldots, \mathbf{c}_k\}$, $\mathbf{c}_j \in \Re^D$, $j = 1, \ldots, k$, coupled with a set of corresponding weight vectors $\{\mathbf{w}_1, \ldots, \mathbf{w}_k\}$, $\mathbf{w}_j \in \Re^D$, $j = 1, \ldots, k$, partition S into k sets:

$$\begin{aligned} S_j &= \{\mathbf{x} | (\sum_{s=1}^{D} w_{js}(x_s - c_{js})^2)^{1/2} \\ &< (\sum_{s=1}^{D} w_{li}(x_s - c_{ls})^2)^{1/2}, \forall l \neq j\}, j = 1, \ldots, k \end{aligned} \tag{2.1}$$

where w_{js} and c_{js} represent the sth components of vectors $\mathbf{w}_j$ and $\mathbf{c}_j$ respectively (ties are broken randomly).

The set of centers and weights is *optimal* with respect to the Euclidean norm, if they minimize the error measure:

$$E_1(C, W) = \sum_{j=1}^{k} \sum_{s=1}^{D} (w_{js} \frac{1}{|S_j|} \sum_{\mathbf{x} \in S_j} (c_{js} - x_s)^2) \tag{2.2}$$

subject to the constraints $\forall j,\ \sum_s w_{js} = 1$. C and W are $(D \times k)$ matrices whose columns are $\mathbf{c}_j$ and $\mathbf{w}_j$ respectively, i.e. $C = [\mathbf{c}_1 \ldots \mathbf{c}_k]$ and $W = [\mathbf{w}_1 \ldots \mathbf{w}_k]$. For mathematical convenience, we set $X_{js} = \frac{1}{|S_j|} \sum_{\mathbf{x} \in S_j} (c_{js} - x_s)^2$, where $|S_j|$ is the cardinality of set S_j. X_{js} represents the average distance from the centroid $\mathbf{c}_j$ of points in cluster j along dimension s. The solution

$$(C^*, W^*) = argmin_{(C,W)} E_1(C, W)$$

will discover one dimensional clusters: it will put maximal (unit) weight on the feature with smallest dispersion X_{js} within each cluster j, and zero weight on all other features. Our objective, instead, is to find weighted multidimensional clusters, where the unit weight gets distributed among all features according to the respective dispersion of data within each cluster. One way to achieve this goal is to add the regularization term $\sum_{s=1}^{D} w_{js} log w_{js}$, which represents the negative entropy of the weight distribution for each cluster. It penalizes solutions with maximal weight on the single feature with smallest dispersion within each cluster. The resulting error function is

$$E_2(C, W) = \sum_{j=1}^{k} \sum_{s=1}^{D} (w_{js} X_{js} + h w_{js} log w_{js}) \tag{2.3}$$

subject to the same constraints $\forall j,\ \sum_s w_{js} = 1$. The coefficient $h \geq 0$ is a parameter of the procedure; it controls the strength of the incentive for clustering on more features. Increasing (decreasing) its value will encourage clusters on more (less) features. This constrained optimization problem can be solved by introducing the

Lagrange multipliers. It gives the solution

$$w_{js}^{*} = \frac{exp(-X_{js}/h)}{\sum_{s=1}^{D} exp(-X_{js}/h)} \tag{2.4}$$

$$c_{js}^{*} = \frac{1}{|S_j|} \sum_{\mathbf{x} \in S_j} x_s \tag{2.5}$$

Solution (2.4) puts increased weights on features along which the dispersion X_{js} is smaller, within each cluster. The degree of this increase is controlled by the value h. Setting $h = 0$, places all weight on the feature s with smallest X_{js}, whereas setting $h = \infty$ forces all features to be given equal weight for each cluster j.

A search strategy needs to be designed to find a partition P that identifies the solution clusters. The authors propose an approach that progressively improves the quality of initial centroids and weights, by investigating the space near the centers to estimate the dimensions that matter the most. *Well-scattered* points in S are first chosen as the k centroids. All weights are initially set to $1/D$. Given the initial centroids $\mathbf{c}_j$, for $j = 1, \ldots, k$, the corresponding sets S_j are computed as previously defined. The average distance X_{js} along each dimension from the points in S_j to $\mathbf{c}_j$ is then computed. The smaller X_{js}, the larger the correlation of points along dimension s. The value X_{js} is used in an exponential weighting scheme to credit weights to features (and to clusters), as given in equation (2.4). The computed weights are used to update the sets S_j, and therefore the centroids' coordinates as given in equation (2.5). The procedure is iterated until convergence is reached. LAC algorithm is summarized in Algorithm 1.

We point out that LAC has shown a highly competitive performance with respect to other state-of-the-art subspace clustering algorithms [19, 20]. However, its

Algorithm 1 LAC Algorithm

Input: n points $\mathbf{x} \in \Re^D$, k, and h.

1. Start with k initial centroids $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_k$
2. Set $w_{sj} = 1/D$, for each centroid $\mathbf{c}_j$, $j = 1, ..., k$ and each feature $s = 1, ..., D$;
3. For each centroids $\mathbf{c}_j$, and for each point $\mathbf{x}$:
   - Set $S_j = \{\mathbf{x} | j = argmin_l L_w(\mathbf{c}_l, \mathbf{x})\}$, where $L_w(\mathbf{c}_l, \mathbf{x}) = (\sum_{s=1}^{D} w_{ls}(c_{ls} - x_s)^2)^{1/2}$;
4. **Compute new weights**: For each centroid $\mathbf{c}_j$, and for each feature s:
   - Set $X_{js} = \sum_{x \in S_j} (c_{js} - x_s)^2 / |S_j|$;
   - Set $w_{js} = \frac{exp(-X_{js}/h)}{\sum_{s=1}^{D} exp(-X_{js}/h)}$
5. For each centroid $\mathbf{c}_j$, and for each point $\mathbf{x}$:
   - Recompute $S_j = \{\mathbf{x} | j = argmin_l L_w(\mathbf{c}_l, \mathbf{x})\}$;
6. **Compute new centroids**. Set $\mathbf{c}_j = \frac{\sum_x x 1_{S_j}(x)}{\sum_x 1_{S_j}(x)}$, for each $j = 1, ..., k$, where $1_S(.)$ is the indicator function of set S;
7. Iterate 3,4,5,6 until convergence.

Output: Set of centroid and weight vectors $\mathbf{c}_j$, $\mathbf{w}_j$ for $j = 1, ..., k$

improvement remains desirable.

The clustering result of LAC depends on two input parameters. The first one is common to all clustering algorithms: the number of clusters k to be discovered in the data. The second one (called h) controls the strength of the incentive to cluster on more features. The setting of h is particularly difficult, since no domain knowledge for its tuning is likely to be available.

Both bottom-up and top-down search methods are useful for analyzing numerical features. However, when the data is defined categorically, a new set of problems

arises.

2.4 Clustering with Categorical Data

2.4.1 Problems and Background

The previous section emphasized techniques for clustering data which are defined in terms of numerical features. Such techniques use inherent geometric properties to define similarity (dissimilarity) between objects to group them. However, when the data is defined by means of categorical features, a new challenge arises. Because each object is described by multiple attributes, none of which can be ordered naturally, attributes' values cannot be ordered in a single way. For example, an attribute of color or shape is not easily ordered [7].

The major issues related to categorical data values are summarized as follows:

- Because their relationship is not linear, categorical attributes have no single ordering. Different orderings of the data lead to different results, none of which can capture the underlying structure of the data.

- Data points which are defined categorically can be described in term of Boolean values $(0, 1)$ for the simple purpose of determining the presence or absence of an object in the itemset. However, because the original categorical values do not have the inherent geometrical structure implied by numerical values, the resulting distance or similarity measures will not yield accurate findings.

Because of these special problems that accrue to the use of distance and similarity measures in categorical data, a new measure is called for, one that is capable of discovering the "natural" grouping of categorical data [7].

Recently, several clustering algorithms for data with categorical attributes have been introduced. The authors in [39] introduced the k-modes algorithm, which is a categorical clustering algorithm that resembles k-means for numerical features. The authors used a new similarity measure that counts the number of mismatches attribute values of pairs of points to update the modes in the clustering process.

Squeezer [68] is a categorical clustering algorithm that redefines similarity-based measurements to assign tuples to clusters. Because categorical values resist attempts to impose geometrical structures on them, the authors instead cast similarity as the number of shared attribute values between a tuple and cluster. The number of shared attribute values between a tuple and a cluster is computed; if the value of this frequency is larger than a predetermined threshold, the tuple is added to the cluster. If the value is lower, the tuple creates a new cluster.

ROCK (Robust Clustering using links) [31] is a hierarchical clustering algorithm for categorical data. It uses the Jaccard coefficient to compute the distance between points. Two points are considered neighbors if their Jaccard similarity exceeds a certain threshold. A link between two points is computed by considering the number of common neighbors. An agglomerative approach is then used to construct the hierarchy.

Another method for clustering categorical data is COOLCAT [12]. Unlike the methods above, which reach their results by applying distance or similarity measures, COOLCAT examines the level of entropy within clusters to accurately group data. As stated by the authors [12], COOLCAT is a more natural method of grouping points, and one that is more closely aligned for categorical data. In addition to these advantages, COOLCAT's entropy-based process is better suited to categorical data than distance- or similarity-based process due to the lack of inherent geometric structures in the data as described above.

2.4.2 COOLCAT

COOLCAT clustering algorithm [12] is a scalable clustering algorithm that discovers clusters with minimal entropy in categorical data. COOLCAT uses categorical, rather than numerical attributes, enabling the mining of real-world datasets offered by fields such as psychology and statistics. The algorithm is based on the idea that a cluster containing similar points has an entropy smaller than a cluster of dissimilar points. Thus, COOLCAT uses entropy to define the criterion for grouping similar objects.

Formally, the entropy measures the uncertainty associated to a random variable. Let X be a random variable with values in $S(X)$, and let $p(x)$ be the corresponding probability function of X. The entropy of X is defined as follows:

$$H(X) = -\sum_{x \in S(X)} p(x) \log(p(x))$$

The entropy of a multivariate vector $\mathbf{X} = (X_1, X_2, \ldots, X_n)$ is defined as:

$$H(\mathbf{X}) = -\sum_{x_1 \in S(X_1)} \cdots \sum_{x_n \in S(X_n)} p(x_1, ..., x_n) \log p(x_1, ..., x_n)$$

.

To minimize the entropy associated to clusters, COOLCAT proceeds as follows. Given n points $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n$, where each point is represented as a vector of D categorical values, $\mathbf{x}_i = (x_i^1, ..., x_i^D)$, COOLCAT partitions the points into k clusters so that the entropy of the clustering is minimized. Let $\hat{C} = \{C_1, \ldots, C_k\}$ represent the

clustering. Then, the entropy associated to $\hat{C}$ is:

$$H(\hat{C}) = \sum_{j=1}^{k} \frac{|C_j|}{n} H(C_j)$$

where $H(C_j)$ is the entropy of cluster C_j:

$$H(C_j) = \sum_{x_i^1 \in S(X^1)} \cdots \sum_{x_i^D \in S(X^D)} P(x_i^1, \ldots, x_i^D | C_j) \log(P(x_i^1, \ldots, x_i^D | C_j))$$

COOLCAT uses an heuristic to incrementally build clusters based on the entropy criterion. It consists of two main phases: an initialization step and an incremental step.

During the initialization phase, COOLCAT bootstraps the algorithm by selecting a sample of points. Out of this sample, it selects the two points that have the maximum pairwise entropy, and so are most dissimilar. COOLCAT places these points in two different clusters. It then proceeds incrementally: at each step, it selects the point that maximizes the minimum pairwise entropy with the previously chosen points. At the end, the k selected points are the initial seeds of the clusters. During the incremental phase, COOLCAT constructs the k clusters. It processes the data in batches. For each data point, it computes the entropy resulting from placing the point in each cluster, and then assigns the point to the cluster that gives the minimum entropy.

The final clustering depends on the order in which points are assigned to clusters, thus there is a danger of obtaining a poor-quality result. To circumvent this problem, the authors of COOLCAT propose a reprocessing step. After clustering a batch of points, a fraction of the set is reconsidered for clustering, where the size of the fraction

is an input parameter. The fraction of points that least fit the corresponding clusters is reassigned to more fitting clusters. To assess which points least match their clusters, COOLCAT counts the number of occurrences of each point's attributes in the cluster to which is assigned. This number of occurrences is then converted into a probability value by dividing it by the cluster size. The point with the lowest probability for each cluster is then removed and reprocessed according to the entropy criterion, as before. By performing this assessment at the conclusion of each incremental step, COOLCAT alleviates the risk imposed by the order of the input of points.

COOLCAT requires four input parameters: the number of clusters k, the sample size used in the initialization step, the buffer size, and the number of points considered for reprocessing. The incremental step of COOLCAT Algorithm summarized in Algorithm 2.

Algorithm 2 Incremental step of COOLCAT Algorithm

1.Given an initial set of clusters $\hat{C} = \{C_1, ..., C_k\}$:
  2.Bring points into memory from disk; for each point $\mathbf{x}$ in memory
    3.For$j = 1, \ldots, k$
      4.Place $\mathbf{x}$ in C_j and compute $H(\hat{C}^j)$
      $\hat{C}^j$ denotes the clustering obtained by placing $\mathbf{x}$ in cluster C_j
      5.Let $t = \arg\min_j(H(\hat{C}^j))$;
    6.Place $\mathbf{x}$ in C_t;
  7.Until all points have been placed in some cluster

Like other single clustering methods, COOLCAT is liable to low accuracy due to the difficulty of tuning input parameters without a cross-validation technique. To compensate for this difficulty, cluster ensembles aggregate results.

2.5 Ensemble Methods

2.5.1 Introduction

In an effort to achieve improved classification accuracy, extensive research has been conducted in classifier ensembles. Ensemble techniques first build a set of base classifiers using a training set, then aggregate the predictions made by the base classifiers to classify the data [62]. In order for classifier ensembles to provide more accurate results than single classifiers, the base classifiers must be both diverse and accurate. Diversity among the classifiers reduces correlated errors. Classifiers that are not diverse produce correlated errors which are carried into the voting process, thereby producing inaccurate results.

Accuracy is required to avoid poor classifiers to obtain a majority of votes. These requirements have been measured. Under simple voting and independent error conditions, if each base classifier has an error rate of less than 50%, then the error rate for ensemble classifiers will decrease monotonically with an increasing number of classifiers [6, 21, 34, 62].

The most well known ensemble techniques are Bagging and Boosting. In both methods, sampling from the original training set is used to generate a set of base classifiers. Bagging [15] uses repeated samples with replacement from the dataset to generate base classifiers. To classify an instance using Bagging, a voting process records votes in each classifier, and an aggregate scheme is applied to select the class with the most votes. Unlike Bagging, Boosting [26] is an iterative procedure. It focuses on misclassified instances, assigning more weight to them in successive iterations and thus changing the distribution of the training set [62].

Classifier ensembles such as Bagging and Boosting have been effective in reducing the generalization error. As the aggregation of classifiers proved to be fruitful, further

research into cluster aggregation has been conducted. From this, cluster ensembles have emerged.

2.5.2 Cluster Ensembles

It is well known that off-the-shelf clustering methods may discover very different structures in a given set of data. This is because each clustering algorithm has its own bias resulting from the optimization of different criteria. Furthermore, there is no ground truth against which the clustering result can be validated. Thus, no cross-validation technique can be carried out to tune input parameters involved in the clustering process. As a consequence, the user is not equipped with any guidelines for choosing the proper clustering method for a given dataset.

In addition to its resistance to cross-validation to tune input parameters, clustering algorithms are dogged by the curse of dimensionality. As the dimensionality of the data increases, clustering algorithms tend to break down. This is due to the likelihood that they will miss important aspects of the clusters' structure in sparse data. Subspace clustering algorithms have been proposed as a method of dealing with the curse of dimensionality, but they, too, have their pitfalls. Specifically, subspace clustering algorithms require input parameters which are not known to the user in advance. Because of these deficiencies, improving upon the performance of both clustering and subspace clustering is desirable.

One feasible solution to this problem is emergent cluster ensembles. Cluster ensembles capture the results produced by different clustering techniques, with the aim of obtaining a new partition that is as good as or better than the best individual clustering. A cluster ensemble can be defined as the process of combining multiple partitions of the dataset into a single partition, with the objective of enhancing the consensus across multiple clustering results. Cluster ensembles can provide robust

and stable solutions by averaging out emergent affected structures due to the various biases to which each participating algorithm is tuned.

2.5.3 Previous work

A cluster ensemble technique is characterized by two components: the mechanism to generate diverse partitions, and the consensus function to combine the input partitions into a final clustering.

Diverse partitions are typically generated by using different clustering algorithms, or by applying a single algorithm with different parameter settings, possibly in combination with data or feature sampling. The k-means algorithm with random initializations [25, 50], or with random number of clusters [49] has been widely used in the literature to generate diverse clusterings. The authors in [64] introduce two techniques, called *weak clustering* algorithms, to produce different partitions. The first technique clusters random one-dimensional projections of multidimensional data; the second one splits the data using random hyperplanes. Random projection is used in [23]. A different approach is proposed in [65], where the ensemble is modeled as a mixture of multivariate multinomial distributions. A unified framework for producing multiple partitions is presented in [63]. [30] applies k-means, k-medoids, and fast *weak clustering* as strategies to generate diversity in clustering results, while [54] proposes a resampling technique that generates and then combines partitions of subsets of the data, to obtain results that reflect the entire dataset.

One popular methodology to build a consensus function utilizes a co-association matrix [25, 30, 54, 64]. An entry value of this matrix represents the average pairwise similarity between points across all input clusterings. Such matrix can be used with any clustering algorithm which operates directly on similarities (e.g., hierarchical clustering) [30, 64]. [50] has shown that good results can be obtained when the

co-association matrix is used as a data matrix in a new feature space, and k-means is ran on it. In alternative to the co-association matrix, voting procedures have been considered to build consensus functions in [65] and in [22]. [28] derives a consensus function based on the Information Bottleneck principle: the mutual information between the consensus clustering and the individual input clusterings is maximized directly, without requiring an approximation.

A different popular mechanism for constructing a consensus maps the problem onto a graph-based partitioning setting [9, 38, 61]. In particular, [61] proposes three graph-based approaches: Cluster-based Similarity Partitioning Algorithm (CSPA), HyperGraph Partitioning Algorithm (HGPA), and Meta-Clustering Algorithm (MCLA). In CSPA, a binary similarity matrix is constructed for each input clustering. Each column corresponds to a cluster: an entry has a value of one if the corresponding point belongs to the cluster, and zero otherwise. An entry-wise average of all the matrices gives an overall similarity matrix, utilized to recluster the data using a graph-partitioning based approach. The induced similarity graph, where vertices correspond to data and edge weights to similarities, is partitioned using METIS [43]. HGPA seeks a partitioning of the hypergraph by cutting a minimal number of hyperedges. (Each hyperedge represents a cluster of an input clustering.) All hyperedges have the same weight. This algorithm looks for a hyperedge separator that partitions the hypergraph into k unconnected components of approximately the same size. It makes use of the package HMETIS [43]. MCLA is based on the clustering of clusters. It provides object-wise confidence estimates of cluster membership. Hyperedges are grouped, and each data point is assigned to the collapsed hyperedge in which it participates most strongly.

Although a considerable amount of research has been conducted in numerical ensemble clustering techniques, less research has been devoted to categorical ensembles.

The authors of [36] generates a partition for each categorical attribute, so that points in each cluster share the same value for that attribute. The resulting clusterings are combined using the consensus functions presented in [61]. The work in [35] constructs cluster ensembles for data with mixed numerical and categorical features.

2.6 Semi-supervised Learning

2.6.1 Introduction

All methods discussed thus far assume that the researcher has no prior knowledge of the data. However, in many learning domains it is relatively easy for the user to provide some information about the domain under study. In areas such as text processing (determining whether two emails are spam or not spam, for example) and image retrieval (determining whether a photograph is of a sporting event, for instance), a human expert can provide some information. For example, the expert can determine whether pairs are similar or dissimilar, and he/she can provide a label for selected data points. Such knowledge is highly valuable and it is capitalizing on this knowledge that attracts researchers to the field of semi-supervised learning.

Semi-supervised learning generally consists of two fields, semi-supervised classification and semi-supervised clustering. In semi-supervised classification, the classic approach to classification, which uses only labeled data, is modified to use both labeled and unlabeled data. Semi-supervised classification techniques use the labeled and unlabeled data to learn models to classify new data. In contrast to this, semi-supervised clustering techniques use the user's knowledge to guide the clustering process of both labeled and unlabeled data [71].

2.6.2 Semi-supervised Classification

Semi-supervised classification uses both labeled and unlabeled data to influence the classification function. Because both types of data are used, more data is available to the classifiers, and a more accurate result is obtained. This distinction is based on the assumption that the unlabeled data hold the same distribution as the labeled data [13]. Examples of semi-supervised classification include self-training, co-training [14], democratic co-learning [69], and tri-training [70].

In self-training, the algorithm employs its own prediction in a reiterative teaching method [58]. The algorithm begins by using a small amount of labeled data to train a given classifier. Next, a number of points with high confidence or high probability of belonging to a certain class, are selected and added along with their anticipated labels to the labeled set. Finally, the classifier retrains itself on this new set, and returns to the unlabeled data. The iterations continue until all unlabeled data are labeled [58]. The pitfall of this method is that an early error in the refinement process will be reinforced in subsequent iterations [71].

Co-training expands on the teaching principle employed in self-training, to include two sub-feature sets and two classifiers. The co-training algorithm assumes that the features can be divided into two independent sets, each of which provides enough information to build a good classifier. As in self-training, the co-training algorithm trains each classifier with labeled data. However, although both classifiers receive the same data, their respective features view it differently. Each classifier classifies unlabeled data, and then supplies the other classifier with the set of points with the highest confidence of accuracy. Finally, each classifier is retrained with the predicted labels provided by its counterpart, and a new round begins. The liability inherent in this method is its assumption of the availability of two independent features within one dataset, a situation that may in fact not occur [14].

Democratic co-learning has two advantages over co-training: first, it can be used with a small amount of labeled data and, second, it does not require two separate feature sets [69]. Democratic co-learning begins by training its multiple classifiers individually on labeled data training sets. A weighted voting procedure is then undertaken for an unlabeled example, and a label is predicted. Because multiple classifiers have been used, it is likely that at least one classifier did not obtain the result predicted by the majority. In this case, the newly labeled example is added to that classifier's training set. This process continues until no further data can be added to the training sets.

Finally, tri-training trains three classifiers using different sets of labeled data. These classifiers predict the labels for unlabeled data through agreement and retraining: if two classifiers agree on the label for a point, this point is added to the training set for the third classifier. This retraining continues until there are no changes in the final results of each classifier. The benefits of tri-training are the avoidance of measuring the confidence of each classifier, and the avoidance of the requirement for independent feature subsets [70].

In our work we extended the co-training and tri-training frameworks to design a semi-supervised clustering ensemble (Chapter 6).

2.6.3 Semi-supervised Clustering

In order to address the ill-posed nature of clustering, researchers have sought to use prior knowledge about the domain to guide the clustering process, thus obtaining clustering results that conform to the user's preference. Because providing labels for clustering algorithms is often prohibitively difficult or expensive, obtaining this kind of knowledge is unrealistic. However, providing pairs of similar and dissimilar examples from a clustering dataset is feasible for the end user. This information is

provided by the end user under the form of must-link and cannot-link constraints on pairs of instances. A must-link constraint indicates that the pair must reside in the same cluster, whereas the cannot-link constraint indicates that the pair must reside in different clusters.

There are two main approaches for semi-supervised clustering, constraint-based and distance-based. In the constraint-based approach, the objective function is modified to satisfy the given constraints. The constraints are enforced during the initialization phase and/or during the iterative phase of the clustering algorithm. In the distance-based approach, the given constraints are used to learn a distance function, which is then used to compute pairwise distances between points.

An example of the constraint-based approach is the COP-Kmeans algorithm [66]. COP-Kmeans is a variation of k-means, where constraints are embedded during the clustering process: each point is assigned to the closest cluster that will enact the least violation of constraints. The algorithm will not assign the point if no such cluster can be found.

Two additional constraint-based variants of k-means are Seeded-KMeans and Constrained-KMeans [13]. In both algorithms, the given labeled data are used to initialize a seeded set; the constraints obtained from this labeled set are then used to guide the k-means algorithm. Seeded-KMeans allows its constraints to be violated in successive iterations, while Constraint-KMeans enforces the constraints in each iteration.

An example of the distance-based approach is the method proposed in [67]. This approach learns positive semi-definite similarity matrices from the given constraints over the input space. The metric learning problem is formulated as a a convex optimization problem which is local-minima-free.

A new approach to semi-supervised data problems was proposed in [18]. The

authors introduced a two-step technique. In the first step, a clustering ensemble approach is designed that uses both labeled and unlabeled points. Then, labeled data points are used to assign clusters to classes. Finally, when a new point arrives the resulting ensemble clustering is used to assign the point to the appropriate cluster, and therefore to the associated class.

Most methods proposed in semi-supervised clustering have focused on developing new clustering algorithms; however, the choice of appropriate constraints has not received enough consideration. In the approach proposed in [29], constraints are imputed from information provided by the co-association values between pairs of points in a clustering ensemble. A co-association value between two points that is below a predetermined threshold becomes a cannot-link constraints; a co-association value above a predetermined threshold becomes a must-link constraint. For co-association values which do not strongly indicate either a must-link or a cannot-link relationship, an oracle expert is queried. These imputed constraints have proven to achieve more accurate results in a semi-supervised clustering method than random constraints. Our research in semi-supervised clustering uses this method of selecting constraints as its basis (see Chapter 6).

Chapter 3: Weighted Clustering Ensembles

3.1 Introduction

As we discussed in Section 2.5.2, clustering is beset by significant challenges. Its difficulties include the question of how to tune input parameters without advance knowledge and the inaccuracies that result in high dimensions.

Our research indicates that clustering ensembles can offer a solution to these challenges.

Definition: Consider a set $S = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ of n points. A clustering ensemble is a collection of m clustering solutions: $G = \{G_1, G_2, ..., G_m\}$. Each clustering solution G_L for $L = 1, \ldots, m$, is a partition of the set S, i.e. $G_L = \{G_L^1, G_L^2, ..., G_L^{K_l}\}$, where $\bigcup_K G_L^K = S$. Given a collection of clustering solutions G and the desired number of clusters k, the objective is to combine the different clustering solutions and compute a new partition of S into k disjoint clusters.

The challenge in cluster ensembles is the design of a proper consensus function that combines the component clustering solutions into an "improved" final clustering. Our methodology in designing a consensus function is to maximize the information shared by multiple partitions of the data and filter out spurious structures discovered by individual clustering algorithms. Designing such a consensus function is not a trivial task. There are three reasons for this. First, the data are unlabeled and little prior knowledge is available. Thus, we cannot design a consensus function based on a voting procedure. Second, a single clustering algorithm can produce several different

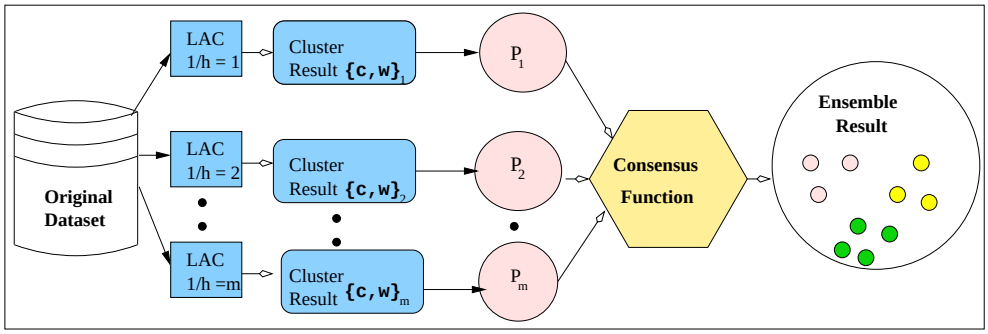

Figure 3.1: The clustering ensemble process

results in several applications to one dataset. Third, several individual clustering algorithms produce multiple results when applied to one dataset. The production of varying results is a difficulty in designing a consensus function because diverse results may not have an explicit correspondence to each other. Despite its difficulties, the clustering ensemble approach can produce results superior to those produced by any single clustering technique, thus benefiting the quality of the research.

3.2 Consensus Functions

As a starting point in designing consensus functions, we introduce the problem of combining multiple weighted clusters, discovered by LAC 2.3.3. Our cluster ensemble methods can be easily extended for any subspace clustering algorithm. Our approach up to this point is the first one that tries to solve the subspace clustering problem through an ensemble function.

For producing multiple clustering results, we focus on setting the parameter h which determines the strength of the incentive for clustering on multiple features. We assume that the number of clusters k is fixed. We leverage the diversity of the clusterings produced by LAC when different values of h are used, in order to generate a consensus clustering that is superior to the participating ones. The major challenge

we face is to find a consensus partition from the outputs of the LAC algorithm to achieve an "improved" overall clustering of the data. Since we are dealing with weighted clusters, we need to design a proper consensus function that makes use of the weight vectors associated with the clusters. Our techniques leverage such weights to define a similarity measure which is associated to the edges of a graph. Then we reduce the problem of defining a consensus function to a graph partitioning problem. This approach has shown good results in the literature [17, 24, 61]. Moreover, the *weighted clusters* computed by the LAC algorithm offer a natural way to define a similarity measure to be integrated in the weights associated to the edges of a graph. The overall clustering ensemble process is illustrated in Figure 3.1.

In the following subsection we introduce our three consensus functions.

3.2.1 Weighted Similarity Partitioning Algorithm (WSPA)

LAC outputs a partition of the data, identified by the two sets $\{\mathbf{c}_1, \ldots, \mathbf{c}_k\}$ and $\{\mathbf{w}_1, \ldots, \mathbf{w}_k\}$. Our aim here is to generate robust and stable solutions via a consensus clustering method. We can generate contributing clusterings by changing the parameter h (as illustrated in Figure 3.1). The objective is then to find a consensus partition from the output partitions of the contributing clusterings, so that an "improved" overall clustering of the data is obtained. Since LAC produces *weighted clusters*, we need to design a consensus function that makes use of the weight vectors associated with the clusters. The details of our approach are as follows.

For each data point $\mathbf{x}_i$, the weighted distance from cluster C_l is given by

$$d_{il} = \sqrt{\sum_{s=1}^{D} w_{ls}(x_{is} - c_{ls})^2}$$

Let $D_i = \max_l\{d_{il}\}$ be the largest distance of $\mathbf{x}_i$ from any cluster. We want to define the probability associated with cluster C_l given that we have observed $\mathbf{x}_i$. At a given point $\mathbf{x}_i$, the cluster label C_l is assumed to be a random variable from a distribution with probabilities $\{P(C_l|\mathbf{x}_i)\}_{l=1}^k$. We provide a nonparametric estimation of such probabilities based on the data and on the clustering result. We do not make any assumption about the specific form (e.g., Gaussian) of the underlying data distributions, thereby avoiding parameter estimations of models, which are problematic in high dimensions when the available data are limited.

In order to embed the clustering result in our probability estimations, the smaller the distance d_{il} is, the larger the corresponding probability credited to C_l should be. Thus, we can define $P(C_l|\mathbf{x}_i)$ as follows:

$$P(C_l|\mathbf{x}_i) = \frac{D_i - d_{il} + 1}{kD_i + k - \sum_l d_{il}} \tag{3.1}$$

where the denominator serves as a normalization factor to guarantee $\sum_{l=1}^k P(C_l|\mathbf{x}_i) = 1$. We observe that $\forall l = 1, \ldots, k$ and $\forall i = 1, \ldots, n$ $P(C_l|\mathbf{x}_i) > 0$. In particular, the added value of 1 in (3.1) allows for a non-zero probability $P(C_L|\mathbf{x}_i)$ when $L = \arg\max_l\{d_{il}\}$. (Any small positive constant achieves this goal, with the normalization factor properly adjusted.) In this last case $P(C_l|\mathbf{x}_i)$ assumes its minimum value $P(C_L|\mathbf{x}_i) = 1/(kD_i + k - \sum_l d_{il})$. For smaller distance values d_{il}, $P(C_l|\mathbf{x}_i)$ increases proportionally to the difference $D_i - d_{il}$: the larger the deviation of d_{il} from D_i, the larger the increase. As a consequence, the corresponding cluster C_l becomes more likely, as it is reasonable to expect based on the information provided by the clustering process. Thus, equation (3.1) provides a nonparametric estimation of the posterior probability associated to each cluster C_l.

We can now construct the vector P_i of posterior probabilities associated with $\mathbf{x}_i$:

$$P_i = (P(C_1|\mathbf{x}_i), P(C_2|\mathbf{x}_i), \ldots, P(C_k|\mathbf{x}_i))^t \tag{3.2}$$

where t denotes the transpose of a vector. The transformation $\mathbf{x}_i \rightarrow P_i$ maps the D dimensional data points $\mathbf{x}_i$ onto a new space of *relative coordinates* with respect to cluster centroids, where each dimension corresponds to one cluster. This new representation embeds information from both the original input data and the clustering result.

To compute the similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$ we used both the cosine similarity and the Kullback-Leibler (KL) divergence. The cosine similarity between probability vectors associated to $\mathbf{x}_i$ and $\mathbf{x}_j$ is defined as:

$$s(\mathbf{x}_i, \mathbf{x}_j) = \frac{P_i^t P_j}{\|P_i\| \|P_j\|} \tag{3.3}$$

In alternative, we compute the distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ using the symmetric KL divergence [47]:

$$d(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{2} \sum_{l=1}^{k} P_{il} \log_2 \frac{P_{il}}{P_{jl}} + \frac{1}{2} \sum_{l=1}^{k} P_{jl} \log_2 \frac{P_{jl}}{P_{il}} \tag{3.4}$$

We then transform the distance into a similarity measure: $s(\mathbf{x}_i, \mathbf{x}_j) = 1 - d(\mathbf{x}_i, \mathbf{x}_j)/(\max_{p,q} d(\mathbf{x}_p, \mathbf{x}_q))$. Both versions of WSPA (with cosine similarity and KL divergence) gave similar results. Thus, we report the results obtained with cosine similarity.

We combine all pairwise similarities (3.3) into an $(n \times n)$ similarity matrix S,

where $S_{ij} = s(\mathbf{x}_i, \mathbf{x}_j)$. We observe that, in general, each clustering may provide a different number of clusters, with different sizes and boundaries. The size of the similarity matrix S is independent of the clustering approach, thus providing a way to align the different clustering results onto the same space, with no need to solve a label correspondence problem.

After running the LAC algorithm m times for different values of the h parameter, we obtain the m similarity matrices $S_1, S_2, \ldots, S_m$. The combined similarity matrix Ψ defines a *consensus function* that can guide the computation of a consensus partition:

$$\Psi = \frac{1}{m} \sum_{l=1}^{m} S_l \tag{3.5}$$

Ψ_{ij} reflects the average similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$ (through P_i and P_j) across the m contributing clusterings.

We now map the problem of finding a consensus partition to a graph partitioning problem. We construct a complete graph $G = (V, E)$, where $|V| = n$ and the vertex V_i identifies $\mathbf{x}_i$. The edge E_{ij} connecting the vertices V_i and V_j is assigned the weight value Ψ_{ij}. We run METIS [43] on the resulting graph to compute a k-way partitioning of the n vertices that minimizes the edge weight-cut [1]. This gives the consensus clustering we seek. The size of the resulting graph partitioning problem is n^2. The steps of the algorithm, which we call WSPA (Weighted Similarity Partitioning Algorithm), are summarized in Algorithm 3.

[1] In our experiments we also apply spectral clustering to compute a k-way partitioning of the n vertices

Algorithm 3 Weighted Similarity Partitioning Algorithm

Input: n points $\mathbf{x} \in R^D$, and k.

1. Run LAC m times with different h values. Obtain m partitions: $\{\mathbf{c}_1^\nu, \ldots, \mathbf{c}_k^\nu\}$,$\{\mathbf{w}_1^\nu, \ldots, \mathbf{w}_k^\nu\}$, $\nu = 1, \ldots, m$
2. For each partition $\nu = 1, \ldots, m$:
   (a) Compute $d_{il}^\nu = \sqrt{\sum_{s=1}^{D} w_{ls}^\nu (x_{is} - c_{ls}^\nu)^2}$
   (b) Set $D_i^\nu = \max_l \{d_{il}^\nu\}$
   (c) Compute $P(C_l^\nu|\mathbf{x}_i) = \frac{D_i^\nu - d_{il}^\nu + 1}{kD_i^\nu + k - \sum_l d_{il}^\nu}$
   (d) Set $P_i^\nu = (P(C_1^\nu|\mathbf{x}_i), P(C_2^\nu|\mathbf{x}_i), \ldots, P(C_k^\nu|\mathbf{x}_i))^t$
   (e) Compute the similarity
   $$s^\nu(\mathbf{x}_i, \mathbf{x}_j) = \frac{P_i^\nu P_j^\nu}{\|P_i^\nu\| \|P_j^\nu\|}, \forall i, j$$
   (f) Construct the matrix S^ν where $S_{ij}^\nu = s^\nu(\mathbf{x}_i, \mathbf{x}_j)$
3. Build the *consensus function* $\Psi = \frac{1}{m} \sum_{\nu=1}^{m} S^\nu$
4. Construct the complete graph $G = (V, E)$, where $|V| = n$ and $V_i \equiv \mathbf{x}_i$. Assign Ψ_{ij} as the weight value of the edge E_{ij} connecting the vertices V_i and V_j
5. Run METIS (or spectral clustering) on the resulting graph G

Output: The resulting k-way partition of the n vertices

3.2.2 Weighted Bipartite Partitioning Algorithm (WBPA)

Our second approach (WBPA) maps the problem of finding a consensus partition to a bipartite graph partitioning problem. This mapping was first introduced in [24]. In [24], however, 0/1 weight values are used. Here we extend the range of weight values to [0,1].

The technique described here has a conceptual advantage with respect to WSPA. We observe that the consensus function ψ used in WSPA measures pairwise similarities which are solely instance-based. On the other hand, the bipartite graph partitioning problem, to which the WBPA technique reduces, partitions both cluster vertices and instance vertices simultaneously. Thus, it also accounts for similarities between clusters. Consider, for example, four instances $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$, and $\mathbf{x}_4$. Suppose that $\mathbf{x}_1$ and $\mathbf{x}_2$ are never clustered together in the input clusterings, and the same holds for $\mathbf{x}_3$ and $\mathbf{x}_4$. However, the groups to which $\mathbf{x}_1$ and $\mathbf{x}_2$ belong often share the same instances, but this is not the case for the groups $\mathbf{x}_3$ and $\mathbf{x}_4$ belong to. Intuitively, we would consider $\mathbf{x}_1$ and $\mathbf{x}_2$ more similar to each other than $\mathbf{x}_3$ and $\mathbf{x}_4$. But WSPA is unable to distinguish these two cases, and may assign low similarity values to both pairs. On the other hand, WBPA is able to differentiate the two cases by modeling both instance-based and cluster-based similarities.

The graph in WBPA models both instances (e.g., data points) and clusters, and the graph edges can only connect an instance vertex to a cluster vertex, forming a bipartite graph. In detail, we proceed as follows for the construction of the graph. Suppose, again, that we run the LAC algorithm m times for different values of the h parameter. For each instance $\mathbf{x}_i$, and for each clustering $\nu = 1, \ldots, m$ we then can compute the vector of posterior probabilities P_i^ν, as defined in equations (3.2) and

(3.1). Using the P vectors, we construct the following matrix A:

$$A = \begin{pmatrix} (P_1^1)^t & (P_1^2)^t & \dots & (P_1^m)^t \\ (P_2^1)^t & (P_2^2)^t & \dots & (P_2^m)^t \\ \vdots & \vdots & & \vdots \\ (P_n^1)^t & (P_n^2)^t & \dots & (P_n^m)^t \\ & & & \end{pmatrix} \tag{3.6}$$

Note that the $(P_i^\nu)^t$s are row vectors (t denotes the transpose). The dimensionality of A is therefore $n \times km$, under the assumption that each of the m clusterings produces k clusters. (We observe that the definition of A can be easily generalized to the case where each clustering may discover a different number of clusters.)

Based on A we can now define a bipartite graph to which our consensus partition problem maps. Consider the graph $G = (V, E)$ with V and E constructed as follows. $V = V^C \cup V^I$, where V^C contains km vertices, each representing a cluster of the ensemble, and V^I contains n vertices, each representing an input data point. Thus $|V| = km + n$. The edge E_{ij} connecting the vertices V_i and V_j is assigned a weight value defined as follows. If the vertices V_i and V_j represent both clusters or both instances, then $E(i, j) = 0$; otherwise, if vertex V_i represents an instance $\mathbf{x}_i$ and vertex V_j represents a cluster C_j^ν (or vice versa) then the corresponding entry of E is $A(i, k(\nu - 1) + j)$. More formally:

- $E(i, j) = 0$ when $((1 \leq i \leq km)$ and $(1 \leq j \leq km))$ or $((km + 1 \leq i \leq km + n)$ and $(km + 1 \leq j \leq km + n))$ (This is the case in which V_i and V_j are both clusters or both instances.)

- $E(i, j) = A(i - km, j)$ when $(km + 1 \leq i \leq km + n)$ and $(1 \leq j \leq km)$ (This is

the case in which V_i is an instance and V_j is a cluster.)

- $E(i,j) = E(j,i)$ when $(1 \leq i \leq km)$ and $(km+1 \leq j \leq km+n)$ (This is the case in which V_i is a cluster and V_j is an instance.)

Note that the dimensionality of E is $(km+n) \times (km+n)$, and E can be written as follows:

$$E = \begin{pmatrix} 0 & A^t \\ A & 0 \end{pmatrix}$$

A partition of the bipartite graph G partitions the cluster vertices and the instance vertices simultaneously. The partition of the instances can then be output as the final clustering. Due to the special structure of the graph G (sparse graph), the size of the resulting bipartite graph partitioning problem is kmn. Assuming that $(km) << n$, this complexity is much smaller than the size n^2 of WSPA.

The steps of the algorithm, which we call WBPA (Weighted Bipartite Partitioning Algorithm), are summarized in Algorithm 4.

We observe that WBPA captures instance-based similarity. Suppose, for example, that $\mathbf{x}_1$ and $\mathbf{x}_2$ are always clustered together in the m input clusterings. Then, the weights, $P(C_i^\nu|\mathbf{x}_1)$ and $P(C_i^\nu|\mathbf{x}_2)$, of the edges connecting $\mathbf{x}_1$ and $\mathbf{x}_2$ to the same cluster vertex C_i^ν have high values, for $\nu = 1, \ldots, m$. As a consequence, the k-way partitioning of the n instances will not cut such edges. As a result, $\mathbf{x}_1$ and $\mathbf{x}_2$ will be grouped together in the final consensus clustering.

Algorithm 4 Weighted Bipartite Partitioning Algorithm

Input: n points $\mathbf{x} \in R^D$, and k

1. Run LAC m times with different h values. Obtain the m partitions: $\{\mathbf{c}_1^\nu, \ldots, \mathbf{c}_k^\nu\}$,$\{\mathbf{w}_1^\nu, \ldots, \mathbf{w}_k^\nu\}$, $\nu = 1, \ldots, m$
2. For each partition $\nu = 1, \ldots, m$:
   (a) Compute $d_{il}^\nu = \sqrt{\sum_{s=1}^{D} w_{ls}^\nu (x_{is} - c_{ls}^\nu)^2}$
   (b) Set $D_i^\nu = \max_l \{d_{il}^\nu\}$
   (c) Compute $P(C_l^\nu | \mathbf{x}_i) = \frac{D_i^\nu - d_{il}^\nu + 1}{kD_i^\nu + k - \sum_l d_{il}^\nu}$
   (d) Set $P_i^\nu = (P(C_1^\nu|\mathbf{x}_i), P(C_2^\nu|\mathbf{x}_i), \ldots, P(C_k^\nu|\mathbf{x}_i))^t$
3. Construct the matrix A as in (3.6)
4. Construct the bipartite graph $G = (V, E)$, where $V = V^C \cup V^I$, $|V^I| = n$ and $V_i^I \equiv \mathbf{x}_i$, $|V^C| = km$ and $V_j^C \equiv C_j$ (a cluster of the ensemble). Set $E(i, j) = 0$ if V_i and V_j are both clusters or both instances. Set $E(i, j) = A(i - km, j) = E(j, i)$ if V_i and V_j represent an instance and a cluster
5. Run METIS (or spectral clustering) on the resulting graph G

Output: The resulting k-way partition of the n vertices in V^I

3.2.3 Weighted Subspace Bipartite Partitioning Algorithm (WSBPA)

The two algorithms WSPA and WBPA provide as output a partition of the data into k clusters, with no information regarding feature relevance for each of the clusters. Here, we discuss a clustering ensemble algorithm (WSBPA) that provides weighted clusters in output. Our approach represents the first attempt in the literature to produce subspace clustering results within the context of ensemble research. This technique advances the WBPA method (3.2.2) by adding to the final partition weighted features associated with each cluster. By assigning a value to each dimension, WSBPA captures the local relevance of features within each cluster. Thus, the structure of the output provided by a single run of LAC is preserved. The output of WSBPA, then, becomes twofold, and has good potential to advance the research on the label assignment problem, which is a difficult and open research issue. For example, for text documents, the analysis of weights assigned to features (i.e., terms) can guide the identification of keywords representative of the topics discussed in the documents. Possibly, relevant keywords, combined with associated weight values, can be used to provide short summaries for clusters and to automatically annotate documents (e.g., for indexing purposes). We will demonstrate this further in Chapter 5.

As we mentioned in our discussion on WBPA, a partition of the bipartite graph G partitions the cluster and the instance vertices simultaneously. However, only the partition of the instance vertices is used to output the final result in WBPA; the partition of the cluster vertices is discarded. WSBPA also uses the partition of cluster vertices; such partition reflects cluster-based similarities. Specifically, WSBPA utilizes the information associated with the partitioned cluster vertices to compute weight vectors for the final clustering.

Let us consider the bipartite graph $G = (V, E)$ as constructed by the algorithm WBPA. We recall that $V = V^C \cup V^I$, where V^C contains km vertices, each representing a cluster of the ensemble, and V^I contains n vertices, each representing an input data point. A k-way partition of the bipartite graph G partitions the cluster vertices and the instance vertices simultaneously into k sets. Furthermore, the k-way partition of G provides a one-to-one correspondence between the k elements of the partition of V^C and the k elements of the partition of V^I. In symbols, let $P_{V^C} = \{V_1^C, V_2^C, \ldots, V_k^C\}$ be the partition of V^C into k sets, and let $P_{V^I} = \{V_1^I, V_2^I, \ldots, V_k^I\}$ be the partition of V^I into k sets. V_j^C and V_j^I, for $j = 1, \ldots, k$, are the sets of cluster vertices and instance vertices grouped together by the k-way partitioning of graph G.

As in WBPA, the partition P_{V^I} provides the resulting clustering of the n input data points $\mathbf{x}_1, \ldots, \mathbf{x}_n$. Each element in P_{V^C} is a set of cluster vertices: $V_l^C = \{v_{l_1}^C, \ldots, v_{l_{|V_l^C|}}^C\}$, for $l = 1, \ldots, k$. Each element in V_l^C represents a cluster from a run of the LAC algorithm. Thus, it has an associated weight vector. Let $\mathbf{w}_{l_i}^C$ be the weight vector associated with the cluster vertex $v_{l_i}^C$. We average the weight vectors $\mathbf{w}_{l_i}^C$, for $i = 1, \ldots, |V_l^C|$, to obtain the weights for cluster V_l^I, for $l = 1, \ldots, k$:

$$\mathbf{w}_l = \frac{1}{|V_l^C|} \sum_{i=1}^{|V_l^C|} \mathbf{w}_{l_i}^C \tag{3.7}$$

We therefore obtain k clusters along with the associated weight vectors: $\{(V_l^I, \mathbf{w}_l)\}_{l=1}^k$. We observe that a k-way partitioning of G that minimizes the edge weight-cut groups together instances $\mathbf{x}$ and clusters C with a high value for $P(C|\mathbf{x})$. This means that, according to LAC clustering, C is a likely cluster given that we have observed $\mathbf{x}$. Thus,

the weight vector for the cluster containing $\mathbf{x}$ should be close to the weight vector associated with C. The averaging in (3.7) gives each cluster C (i.e., the corresponding weight) with high $P(C|\mathbf{x})$ equal importance for the computation of the weight of the cluster containing $\mathbf{x}$. The steps of the algorithm, which we call Weighted Subspace Bipartite Partitioning Algorithm (WSBPA) are summarized in the following.

Algorithm 5 Weighted Subspace Bipartite Partitioning Algorithm

Input: n points $\mathbf{x} \in R^D$, and k

1. Run LAC m times with different h values. Obtain the m partitions: $\{\mathbf{c}_1^\nu, \ldots, \mathbf{c}_k^\nu\}$,$\{\mathbf{w}_1^\nu, \ldots, \mathbf{w}_k^\nu\}$, $\nu = 1, \ldots, m$
2. For each partition $\nu = 1, \ldots, m$:
   (a) Compute $d_{il}^\nu = \sqrt{\sum_{s=1}^{D} w_{ls}^\nu (x_{is} - c_{ls}^\nu)^2}$
   (b) Set $D_i^\nu = \max_l \{d_{il}^\nu\}$
   (c) Compute $P(C_l^\nu|\mathbf{x}_i) = \frac{D_i^\nu - d_{il}^\nu + 1}{kD_i^\nu + k - \sum_l d_{il}^\nu}$
   (d) Set $P_i^\nu = (P(C_1^\nu|\mathbf{x}_i), P(C_2^\nu|\mathbf{x}_i), \ldots, P(C_k^\nu|\mathbf{x}_i))^t$
3. Construct the A matrix as in (3.6)
4. Construct the bipartite graph $G = (V, E)$ as in the algorithm WBPA
5. Run METIS (or spectral clustering) on the resulting graph G. Consider the resulting partitions $P_{V^C} = \{V_1^C, V_2^C, \ldots, V_k^C\}$ and $P_{V^I} = \{V_1^I, V_2^I, \ldots, V_k^I\}$ of the cluster and instance vertices respectively
6. Compute the average weight vector $\mathbf{w}_l$ for each element V_l^C in P_{V^C}, as given in equation (3.7)

Output: The resulting weight vectors coupled with the corresponding cluster centroids: $\{(\mathbf{c}_l^I, \mathbf{w}_l)\}_{l=1}^k$, where $\mathbf{c}_l^I$ is the centroid of cluster V_l^I

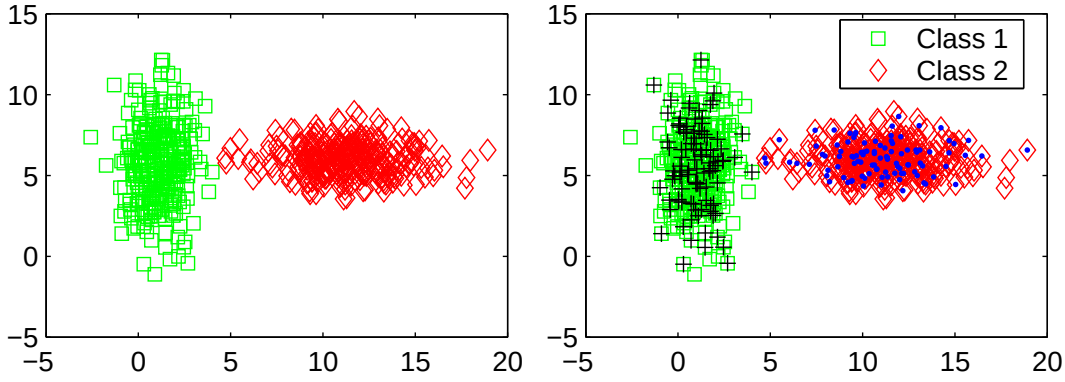

Figure 3.2: (Left): Two-Gaussian data. (Right): Random sampling of 100 points (crosses and dots) from each cluster.

3.3 An Illustrative Example

Here we present and discuss an illustrative example to demonstrate that the relative coordinates $P(C|\mathbf{x})$ provide a suitable representation for the computation of pairwise similarities. We emphasize that this is an important point since the information provided by the subspace clustering is embedded into these coordinates, and, in turn, the proposed consensus function is constructed upon such representation of the data. Thus, the efficacy of the consensus function itself relies on the suitability of these coordinates.

We have designed one simulated dataset with two clusters distributed as bivariate Gaussians (Figure 3.2 (Left)). The mean and standard deviation vectors for each cluster are as follows: $\mathbf{m}_1 = (0.5, 5)$, $\mathbf{s}_1 = (1, 9)$; $\mathbf{m}_2 = (12, 5)$, $\mathbf{s}_2 = (6, 2)$. Each cluster has 300 points. We ran the LAC algorithm on the Two-Gaussian dataset for two values of the $1/h$ parameter (7 and 12). For $(1/h) = 7$, LAC provides a perfect separation (the error rate is 0.0%); the corresponding weight vectors associated to each cluster are $\mathbf{w}_1^{(7)} = (0.81, 0.19)$, $\mathbf{w}_2^{(7)} = (0.18, 0.82)$. For $(1/h) = 12$, the error rate of LAC is 5.3%; the weight vectors in this case are $\mathbf{w}_1^{(12)} = (0.99, 0.01)$,

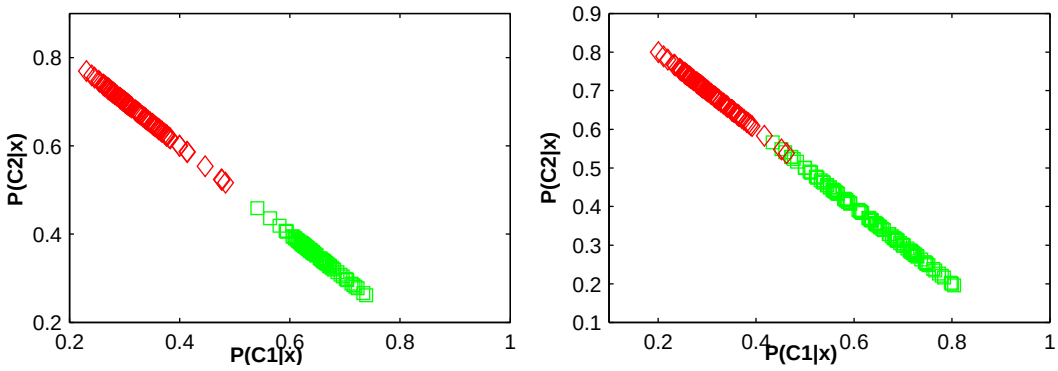

Figure 3.3: (Left): Two dimensional probability vectors $P = (P(C_1|\mathbf{x}), P(C_2|\mathbf{x}))^t$, $(1/h) = 7$. LAC error rate is 0.0%. (Right): Two dimensional probability vectors $P = (P(C_1|\mathbf{x}), P(C_2|\mathbf{x}))^t$, $(1/h) = 12$.

$\mathbf{w}_2^{(12)} = (0.0002, 0.9998)$.

For the purpose of plotting the two-dimensional posterior probability vectors associated with each point $\mathbf{x}$, we consider a random sample of 100 points from each cluster (as shown in Figure 3.2 (Right)). The probability vectors (computed as in equations (3.2) and (3.1)) of such sample points are plotted in Figure 3.3 (Left) and Figure 3.3 (Right), respectively for $(1/h) = 7$ and $(1/h) = 12$. We observe that in Figure 3.3 (Left) ($(1/h) = 7$) for points $\mathbf{x}$ of cluster 1 (green points square-shaped) $P(C_1|\mathbf{x}) > P(C_2|\mathbf{x})$, and for points $\mathbf{x}$ of cluster 2 (red points diamond-shaped) $P(C_2|\mathbf{x}) > P(C_1|\mathbf{x})$. Thus, there is no overlapping (in relative coordinate space) between points of the two clusters, and LAC achieves a perfect separation (the error rate is 0.0%). On the other hand, Figure 3.3 (Right) ($(1/h) = 12$) demonstrates that for a few points $\mathbf{x}$ of cluster 1 (green points square-shaped) $P(C_1|\mathbf{x}) < P(C_2|\mathbf{x})$ (overlapping region in Figure 3.3 (Right)). LAC misclassifies these points as members of cluster 2, which results in an error rate of 5.3%.

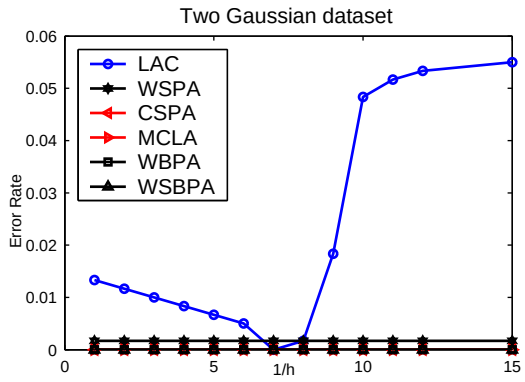

Figure 3.4: Results on Two-Gaussian data. METIS was used in conjunction with WSPA, WBPA, and WSBPA.

Thus, the relative coordinates $P(C|\mathbf{x})$ provide a suitable representation to compute the pairwise similarity measure in our clustering ensemble approaches. By combining the clustering results in the relative coordinate space obtained by different runs of LAC, we aim at utilizing the consensus across multiple clusterings, while averaging out emergent spurious structures. The experimental results obtained for this dataset (presented in the next Chapter) corroborate our analysis. In fact, we anticipate here that our three clustering ensemble methods WSPA, WBPA, and WSBPA achieved 0.17%, 0.0%, and 0.0% error rates, respectively. Thus, they successfully separated the two clusters, as the best input clustering provided by LAC did (see Table 4.3 and Figure 3.4 for details).

Chapter 4: Experimental Results

4.1 Experimental Design and Results

We have designed two simulated datasets to analyze the behavior of the proposed techniques in a controlled setting. These datasets contain two and three clusters, respectively, distributed as bivariate Gaussians (Figures 3.2 (Left) and 4.1). The mean and standard deviation vectors for the Two-Gaussian dataset are as described in Section 3.3. The mean and standard deviation vectors for the Three-Gaussian dataset are as follows: $\mathbf{m}_1 = (2, 5)$, $\mathbf{s}_1 = (1, 9)$; $\mathbf{m}_2 = (12, 5)$, $\mathbf{s}_2 = (6, 2)$; $\mathbf{m}_3 = (23, 5)$, $\mathbf{s}_3 = (1, 9)$. In our experiments, we also used seven real datasets. The characteristics of all datasets are given in Table 5.3. Iris, Breast, Letter(A,B), and SatImage are from the UCI Machine Learning Repository [8]. WDBC is the Wisconsin Diagnostic Breast Cancer dataset [52]. Spam2000 and Spam5996 are two high dimensional text (spam) datasets. The documents in each dataset were preprocessed by eliminating stop words (based on a stop words list) and stemming words to their root source. As feature values in the vector space model we have used the frequency of the terms in the corresponding document. Both Spam2000 and Spam5996 belong to the Email-1431 dataset[1]. This dataset consists of emails falling into three categories: conference (370), jobs (272), and spam (786). We ran two different experiments with this dataset. In one case we reduced the dimensionality to 2000 terms (Spam2000), and in the second case to 5996 (Spam5996). In both cases we consider two clusters by merging

[1]The Email-1431 dataset was created by Finn Arup Nielsen. It is available at: http://www.imm.dtu.dk/∼rem/data/Email-1431.zip

Table 4.1: Characteristics of the datasets

Dataset	k	D	n (points-per-class)
Two-Gaussian	2	2	600 (300-300)
Three-Gaussian	3	2	900 (300-300-300)
Iris	3	4	150 (50-50-50)
WDBC	2	31	424 (212-212)
Breast	2	9	478 (239-239)
Letter(A,B)	2	16	1555 (789-766)
SatImage	2	36	2110 (1072-1038)
Spam2000	2	2000	1284 (642- 642)
Spam5996	2	5996	1284 (642- 642)

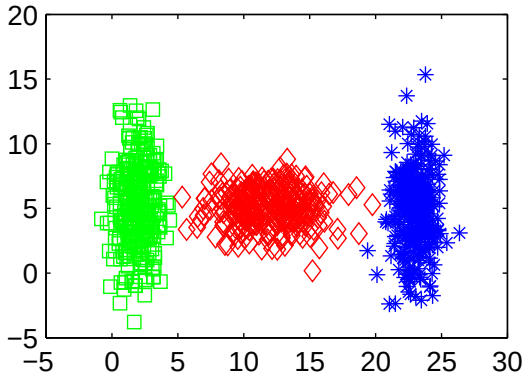

Figure 4.1: Three Gaussian dataset

the conference and jobs mails into one group (non-spam).

Since METIS [43] requires balanced datasets, we performed random sampling on Breast, WDBC, Spam2000 and Spam5996. In each case, we sub-sampled the most populated class: from 357 to 212 for WDBC, from 444 to 239 for Breast , and from 786 to 642 for Spam2000 and Spam5996. For the Letter dataset, we used the classes "A" and "B" (balanced), and for the SatImage we used classes 1 and 7 (again balanced).

Besides METIS, we also used spectral clustering[2] [55] to compute the k-way partitioning of the resulting graph, for the three techniques WSPA, WBPA, and WSBPA.

[2]We used the Matlab Toolbox available at: http://www.cs.washington.edu/homes/sagarwal/code.html.

The advantage of spectral clustering over METIS is that spectral clustering does not require balanced data. Here, for comparison purposes, we apply both METIS and spectral clustering on the same balanced data. Our objective is to demonstrate the applicability of spectral clustering in conjunction with our ensemble techniques, thus enabling the use of our methods also with unbalanced data.

We compared our weighted clustering ensemble techniques (WSPA, WBPA, and WSBPA) with the three methods CSPA, HGPA, and MCLA [61]. Like our methods, these three techniques transform the problem of finding a consensus clustering into a graph partitioning problem, and make use of METIS. Thus, it was a natural choice for us to compare our methods with these approaches. We consider the partitions provided by LAC (and discard the weights) in order to run CSPA, HGPA, and MCLA, since these methods are designed to accept clusterings (not subspace clusterings). Here, we report the accuracy achieved by CSPA and MCLA, as HGPA was consistently the worst. The ClusterPack Matlab Toolbox was used[3].

To further analyzing the benefits of diverse results generated by means of subspace clustering, we also considered a consensus function not based on a graph partitioning problem. The specific goals of these experiments are: (1) Test whether the diverse clusterings produced by LAC can be effectively combined using a consensus function based on a co-association matrix; and (2) compare our approach of generating diversity with alternate approaches available in the literature (e.g., varying k-means). To this end, we ran LAC with different values of h as before. For each of the m resulting partitions (weights are discarded), we construct a co-association matrix T of size $n \times n$, where $T_{ij}^{(l)} = 1$ if x_i and x_j are clustered together in partition l, $T_{ij}^{(l)} = 0$ otherwise. A final co-association matrix T is derived by averaging the individual $T^{(l)}$, $l = 1, \ldots, m$: $T_{ij} = \frac{1}{m} \sum_{l=1}^{m} T_{ij}^{(l)}$, $i, j = 1, \ldots, n$. Previous work [51,57] has shown that

[3]Available at: www.lans.ece.utexas.edu/∼strehl/

good results can be obtained when T is used as a data matrix in a new feature space (rather then a similarity matrix). Thus, we used T as data, and ran k-means on it [51]. We identify the resulting method as LAC+Co-as. To account for the subspace structure discovered by LAC, we also consider Ψ (as defined in (3.5)) as data matrix. We call this approach LAC+wCo-as. In addition, we ran the same consensus function on clusterings generated by k-means with random initializations as well. The resulting approach is denoted as k-means+Co-as. We observe that the consensus function has a random element (as it relies on k-means). Thus, we ran it 10 times, and report average accuracies.

Evaluating the quality of clustering is in general a difficult task. Since class labels are available for the datasets used here, we evaluate the results by computing the error rate and the normalized mutual information (NMI). The error rate is computed according to the confusion matrix. The NMI provides a measure that is impartial with respect to the number of clusters [61]. It reaches its maximum value of one only when the result completely matches the original labels. The NMI is computed according to the average mutual information between every pair of cluster and class [61]:

$$NMI = \frac{\sum_{i=1}^{k}\sum_{j=1}^{k} n_{i,j}\log\frac{n_{i,j}n}{n_i n_j}}{\sqrt{\sum_{i=1}^{k} n_i \log\frac{n_i}{n}\sum_{j=1}^{k} n_j \log\frac{n_j}{n}}} \tag{4.1}$$

where $n_{i,j}$ is the number of agreement between cluster i and class j, n_i is the number of data in cluster i, n_j is the number of data in class j, and n is the total number of points.

We observe that the algorithm WSBPA outputs weight vectors coupled with the corresponding cluster centroids: $\{(\mathbf{c}_l^I, \mathbf{w}_l)\}_{l=1}^{k}$. In order to compute the corresponding

partition, we assign each point to the closest centroid according to the locally weighted Euclidean distance.

4.2 Analysis of the Results

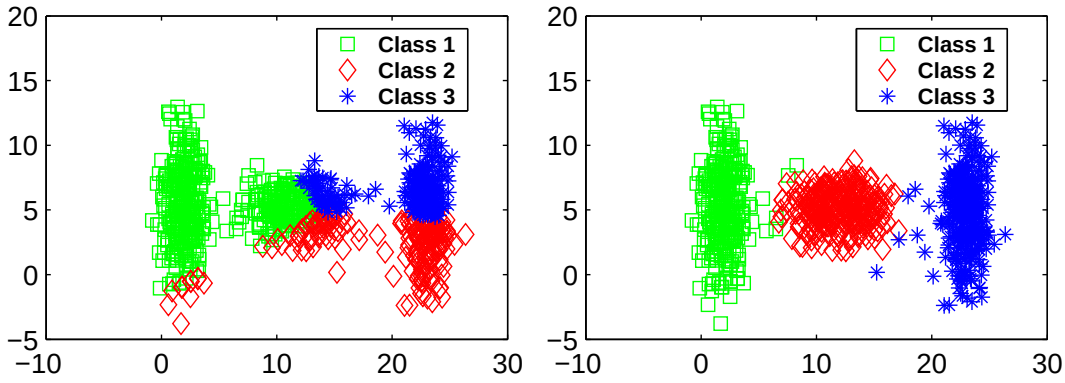

Figure 4.2: (Left): LAC: Clustering results for Three-Gaussian data, $(1/h) = 1$. The error rate is 34.6%. (Right): LAC: Clustering results for Three-Gaussian data, $(1/h) = 4$. The error rate is 1.3%

For each dataset, we ran the LAC algorithm for several values of the input parameter h. The clustering results of LAC are then given as input to the consensus clustering techniques being compared. (As the value of k, we input both LAC and the ensemble algorithms with the actual number of classes in the data.) Figures 3.4 and 4.3 plot the error rate (%) achieved by LAC as a function of the $1/h$ parameter, for each dataset considered. The error rates of our weighted clustering ensemble methods (WSPA, WBPA, and WSBPA in conjunction with METIS), and of the CSPA and MCLA techniques are also reported. Each figure clearly shows the sensitivity of the LAC algorithm to the value of h. The trend of the error rate clearly depends on the data distribution. Detailed results for all data are provided in Tables 4.3-4.11, where we report the NMI and error rate (ER) of the ensembles, and the maximum, minimum,

and average NMI and error rate values for the input clusterings. Eleven methods are being compared: our three methods WSPA, WBPA, WSBPA, each combined with both METIS and spectral clustering (SPEC is short for spectral clustering), CSPA and MCLA, and the three techniques based on a co-association matrix.

We further illustrate the sensitivity of the LAC algorithm to the value of h for the Three-Gaussian data (Figure 4.1). Figures 4.2 (Left) and 4.2 (Right) depict the clustering results of LAC for $(1/h) = 1$ and $(1/h) = 4$, respectively. Figure 4.2 (Left) clearly shows that for $(1/h) = 1$, LAC is unable to discover the structure of the three clusters, and gives an error rate of 34.6%. On the other hand, LAC achieves a nearly perfect separation for $(1/h) = 4$, as shown in Figure 4.2 (Right). The error rate in this case is 1.3%, which is also the minimum achieved in all the runs of the algorithm. Results for the ensemble techniques on the Three-Gaussian data are given in Figure 4.3 and in Table 4.4. We observe that the WSPA(-METIS) technique perfectly separates the data (0.0% error), and that WBPA(-METIS) gives a 0.44% error rate. In both cases, the error rate achieved is lower than the minimum error rate among the input clusterings (1.3%). Moreover, WSBPA gives an error rate of 1.3%, which is equal to the lowest error rate achieved by LAC. We note that WSBPA(-METIS) and MCLA provide the same error rate for this problem. However, WSBPA produces not only a partition of points as the final result, but also relevance values of features associated with each cluster. In this regard, WSBPA provides more information, and is therefore superior to MCLA.

In general, all three our ensemble techniques were able to filter out spurious structures identified by individual runs of LAC, and provided a better error rate than (or equal to) LAC's minimum error rate. For all seven real datasets either WBPA, WSPA, or WSBPA provided the lowest error rate among the methods being compared. For the Iris, WDBC, Breast, SatImage, and Spam5996 datasets (five out of seven total),

the error rate provided by the WBPA technique is as good or better than the best individual input clustering. For the Letter(A,B) and Spam2000 datasets, the error rate of WBPA is still below the average error rate of the input clusterings. WSPA gave excellent results as well. For Iris, WDBC, Breast, SatImage, and Spam5996 the error rate provided by WSPA is lower than the best individual input clustering. For Spam2000 (with METIS) and Letter(A,B) the error rate of WSPA is well below the average error rate of the input clusterings.

Also WSBPA performed quite well. It produced error rates comparable with, and sometime better than, the other techniques. In addition, WSBPA provides information on the relevance of features associated with each cluster. In each dataset, WSBPA achieved a result far superior to the average error rate of the input clusterings. Furthermore, we note that for Iris, SatImage, Spam2000, and Spam5996 (four out of seven total) WSBPA has provided a result superior to both the results provided by CSPA and MCLA. In particular, WSBPA (both with METIS and SPEC) produced excellent results for the high dimensional data Spam2000 and Spam5996. In these two cases, WSBPA produced better results than the four competing techniques, and achieved a lower error rate than (or equal to) the minimum error rate among the input clusterings.

Clearly, our weighted clustering ensemble techniques are capable of achieving superior accuracy results with respect to the CSPA and MCLA techniques on the tested datasets. This result is summarized in Table 4.2, where we report the average NMI and average error rate on all real datasets. We observe that, on average, SPEC performed better than METIS. We also report the average values for the LAC algorithm to emphasize the large improvements obtained by the ensembles across the real datasets. Given the competitive behavior shown by LAC in the literature [19], this is a significant result.

Table 4.2: Average NMIs and error rates

	Avg-NMI	Avg-Error
WSPA-METIS	0.693	7.07
WSPA-SPEC	**0.729**	**6.04**
WBPA-METIS	0.705	6.67
WBPA-SPEC	0.728	6.14
WSBPA-METIS	0.661	8.51
WSBPA-SPEC	0.669	8.32
CSPA	0.655	8.59
MCLA	0.647	9.36
LAC	0.576	13.59
LAC+Co-as.	0.646	10.07
LAC+wCo-as.	0.649	9.7
k-means+Co-as.	0.540	16.78

We observe that the consensus function Ψ defined in (3.5) measures the similarity of points in terms of how close the "patterns" captured by the corresponding probability vectors are. As a consequence, Ψ (as well as the matrix A for the WBPA and WSBPA techniques) takes into account not only how often the points are grouped together across the various input clusterings, but also the degree of confidence of the groupings. On the other hand, the CSPA and MCLA approaches take as input the partitions provided by each contributing clustering algorithm. That is, $\forall \nu$ and $\forall i$, $P(C_l^\nu|\mathbf{x}_i) = 1$ for a given l, and 0 otherwise. Thus, the information concerning the degree of confidence associated with the clusterings is lost. This is likely the reason for the superior performance achieved by our weighted clustering ensemble algorithms.

In some cases, the WBPA technique gives a lower error rate compared to the WSPA technique (WBPA-METIS performs slightly better than WSPA-METIS, on average). This result may be due to the conceptual advantage of WBPA with respect to WSPA discussed at the beginning of Section 3.2.2. The consensus function ψ used in WSPA measures pairwise similarities which are solely instance-based. On the other hand, the bipartite graph partitioning problem, to which the WBPA technique

Table 4.3: Results on Two-Gaussian data

	Ens-NMI	Ens-ER	Max-NMI	Min-NMI	Avg-NMI	Max-ER	Min-ER	Avg-ER
WSPA-METIS	0.984	0.17	1	0.75	0.88	5.5	0	2.2
WSPA-SPEC	0.911	1.3	1	0.75	0.88	5.5	0	2.2
WBPA-METIS	**1**	**0**	1	0.75	0.88	5.5	0	2.2
WBPA-SPEC	0.911	1.3	1	0.75	0.88	5.5	0	2.2
WSBPA-METIS	**1**	**0**	1	0.75	0.88	5.5	0	2.2
WSBPA-SPEC	**1**	**0**	1	0.75	0.88	5.5	0	2.2
CSPA	**1**	**0**	1	0.75	0.88	5.5	0	2.2
MCLA	**1**	**0**	1	0.75	0.88	5.5	0	2.2
LAC+Co-as.	**1**	**0**	1	0.75	0.88	5.5	0	2.2
LAC+wCo-as.	0.983	0.18	1	0.75	0.88	5.5	0	2.2
k-means+Co-as.	0.91	1.3	0.91	0.91	0.91	1.3	1.3	1.3

Table 4.4: Results on Three Gaussian data

	Ens-NMI	Ens-ER	Max-NMI	Min-NMI	Avg-NMI	Max-ER	Min-ER	Avg-ER
WSPA-METIS	**1**	**0**	0.940	0.376	0.789	34.9	1.3	10.5
WSPA-SPEC	0.912	2.2	0.940	0.376	0.789	34.9	1.3	10.5
WBPA-METIS	0.976	0.44	0.940	0.376	0.789	34.9	1.3	10.5
WBPA-SPEC	0.940	1.3	0.940	0.376	0.789	34.9	1.3	10.5
WSBPA-METIS	0.940	1.3	0.940	0.376	0.789	34.9	1.3	10.5
WSBPA-SPEC	0.933	1.56	0.940	0.376	0.789	34.9	1.3	10.5
CSPA	0.893	2.3	0.940	0.376	0.789	34.9	1.3	10.5
MCLA	0.940	1.3	0.940	0.376	0.789	34.9	1.3	10.5
LAC+Co-as.	0.795	17.3	0.940	0.376	0.789	34.9	1.3	10.5
LAC+wCo-as.	0.899	2.7	0.940	0.376	0.789	34.9	1.3	10.5
k-means+Co-as.	0.834	17.2	0.949	0.945	0.946	1.2	1.1	1.17

reduces, partitions both cluster vertices and instance vertices simultaneously. Thus, it also accounts for similarities between clusters.

The results obtained for LAC+Co-as. and LAC+wCo-as. show that the diverse clusterings produced by LAC can be effectively combined using also a consensus function based on a co-association matrix. LAC+wCo-as. gives on average lower error rates than LAC+Co-as. This is expected since LAC+wCo-as. embeds the subspace structure discovered by LAC into the consensus function. The co-association matrix is also effective when combined with k-means (note that the high error rate of k-means+Co-as. on the WDBC data is due to the fact that k-means gave the same

Table 4.5: Results on Iris data

	Ens-NMI	Ens-ER	Max-NMI	Min-NMI	Avg-NMI	Max-ER	Min-ER	Avg-ER
WSPA-METIS	0.744	10.00	0.758	0.657	0.709	17.3	9.3	12.9
WSPA-SPEC	**0.824**	**6.00**	0.758	0.657	0.709	17.3	9.3	12.9
WBPA-METIS	0.754	9.3	0.758	0.657	0.709	17.3	9.3	12.9
WBPA-SPEC	0.813	6.6	0.758	0.657	0.709	17.3	9.3	12.9
WSBPA-METIS	0.727	11.3	0.758	0.657	0.709	17.3	9.3	12.9
WSBPA-SPEC	0.774	9.3	0.758	0.657	0.709	17.3	9.3	12.9
CSPA	0.677	13.3	0.758	0.657	0.709	17.3	9.3	12.9
MCLA	0.708	13.3	0.758	0.657	0.709	17.3	9.3	12.9
LAC+Co-as.	0.654	20.8	0.758	0.657	0.709	17.3	9.3	12.9
LAC+wCo-as.	0.715	15.7	0.758	0.657	0.709	17.3	9.3	12.9
k-means+Co-as.	0.699	19.9	0.758	0.590	0.705	33.3	10.6	17.6

Table 4.6: Results on WDBC data

	Ens-NMI	Ens-ER	Max-NMI	Min-NMI	Avg-NMI	Max-ER	Min-ER	Avg-ER
WSPA-METIS	0.512	10.6	0.524	0.009	0.329	48.5	11.1	23.4
WSPA-SPEC	0.521	10.3	0.524	0.009	0.329	48.5	11.1	23.4
WBPA-METIS	**0.573**	**8.7**	0.524	0.009	0.329	48.5	11.1	23.4
WBPA-SPEC	0.521	10.3	0.524	0.009	0.329	48.5	11.1	23.4
WSBPA-METIS	0.482	12.5	0.524	0.009	0.329	48.5	11.1	23.4
WSBPA-SPEC	0.480	12.7	0.524	0.009	0.329	48.5	11.1	23.4
CSPA	0.498	11.1	0.524	0.009	0.329	48.5	11.1	23.4
MCLA	0.457	13.4	0.524	0.009	0.329	48.5	11.1	23.4
LAC+Co-as.	0.464	12.9	0.524	0.009	0.329	48.5	11.1	23.4
LAC+wCo-as.	0.469	12.7	0.524	0.009	0.329	48.5	11.1	23.4
k-means+Co-as.	0.0005	49.7	0.0005	0.0005	0.0005	49.7	49.7	49.7

high error rate on each single run. See Table 4.6.) Overall, though, LAC provides better accuracy/diversity trade-offs, which lead to more accurate ensembles (see Table 4.2).

We finally tested how the size of the ensemble affects the error rate. Figure 4.4 shows the results for WSPA-METIS and WBPA-METIS on the real data sets. Each point corresponds to an average of ten ensembles of the corresponding size. Ensemble components are randomly chosen from a collection of 50 partitions obtained by running LAC with $1/h = 1, \ldots, 50$. Ensemble sizes between 10 and 45 are considered. Overall, the error rate slowly decreases as the ensemble size increases. An ensemble

Table 4.7: Results on Breast data

	Ens-NMI	Ens-ER	Max-NMI	Min-NMI	Avg-NMI	Max-ER	Min-ER	Avg-ER
WSPA-METIS	**0.779**	**3.6**	0.700	0.197	0.422	34.1	5.9	20.5
WSPA-SPEC	0.772	3.77	0.700	0.197	0.422	34.1	5.9	20.5
WBPA-METIS	**0.779**	**3.6**	0.700	0.197	0.422	34.1	5.9	20.5
WBPA-SPEC	0.772	3.77	0.700	0.197	0.422	34.1	5.9	20.5
WSBPA-METIS	0.585	9.6	0.700	0.197	0.422	34.1	5.9	20.5
WSBPA-SPEC	0.574	10.0	0.700	0.197	0.422	34.1	5.9	20.5
CSPA	0.722	4.8	0.700	0.197	0.422	34.1	5.9	20.5
MCLA	0.575	10.3	0.700	0.197	0.422	34.1	5.9	20.5
LAC+Co-as.	0.624	8.2	0.700	0.197	0.422	34.1	5.9	20.5
LAC+wCo-as.	0.561	11.0	0.700	0.197	0.422	34.1	5.9	20.5
k-means+Co-as.	0.722	5.2	0.737	0.722	0.728	5.2	4.8	5.1

Table 4.8: Results on Letter(A,B) data

	Ens-NMI	Ens-ER	Max-NMI	Min-NMI	Avg-NMI	Max-ER	Min-ER	Avg-ER
WSPA-METIS	0.579	8.6	0.707	0.001	0.514	47.9	6.4	13.6
WSPA-SPEC	**0.698**	**6.6**	0.707	0.001	0.514	47.9	6.4	13.6
WBPA-METIS	0.592	8.2	0.707	0.001	0.514	47.9	6.4	13.6
WBPA-SPEC	**0.698**	**6.6**	0.707	0.001	0.514	47.9	6.4	13.6
WSBPA-METIS	0.537	9.9	0.707	0.001	0.514	47.9	6.4	13.6
WSBPA-SPEC	0.551	9.4	0.707	0.001	0.514	47.9	6.4	13.6
CSPA	0.579	8.6	0.707	0.001	0.514	47.9	6.4	13.6
MCLA	0.512	10.8	0.707	0.001	0.514	47.9	6.4	13.6
LAC+Co-as.	0.512	10.8	0.707	0.001	0.514	47.9	6.4	13.6
LAC+wCo-as.	0.534	10.0	0.707	0.001	0.514	47.9	6.4	13.6
k-means+Co-as.	0.512	11.9	0.658	0.321	0.489	18.0	7.3	12.6

size of 25-30 components seems to be a reasonable choice in general.

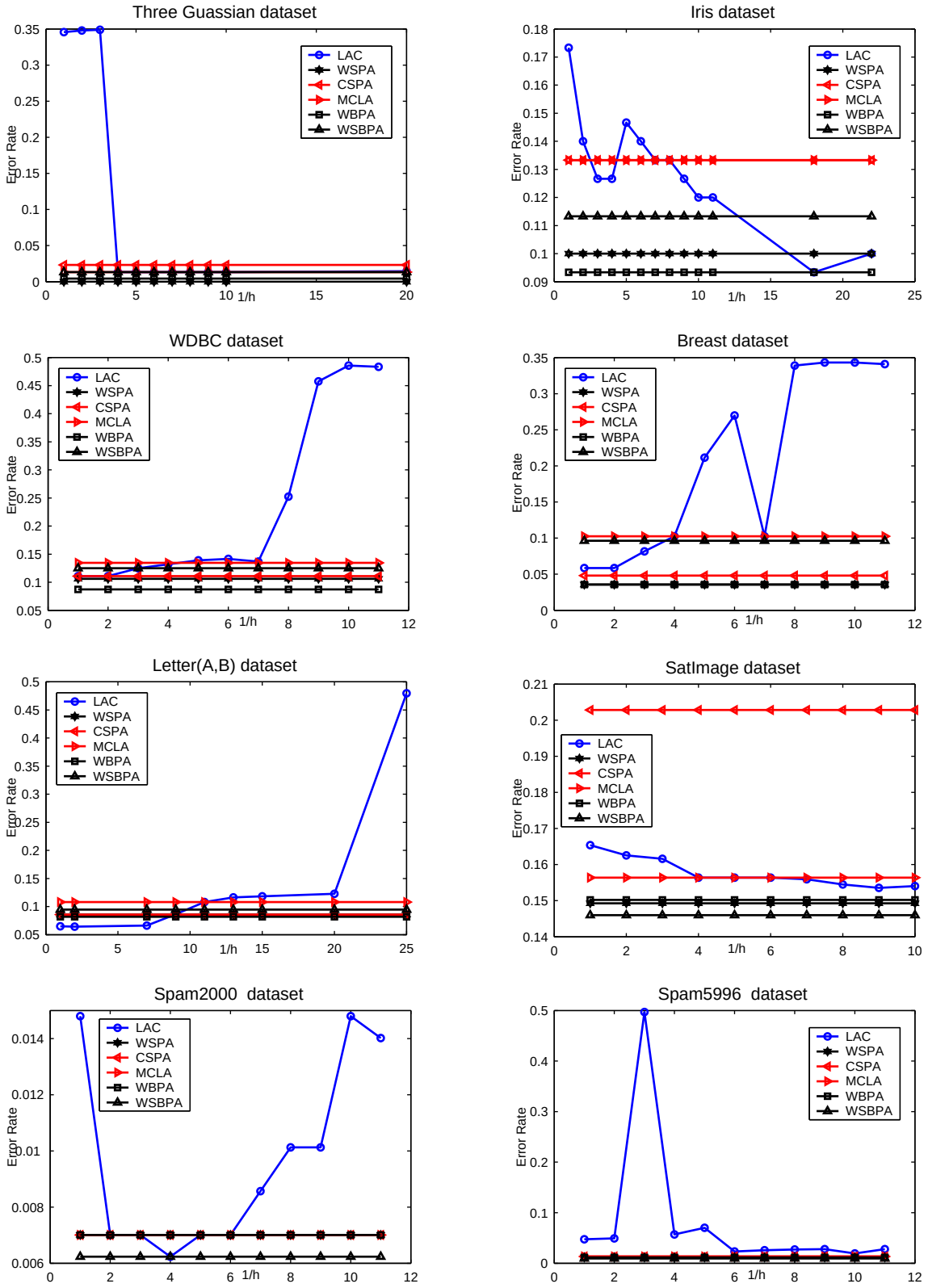

Figure 4.3: Clustering Ensemble Results. METIS was used in conjunction with WSPA, WBPA, and WSBPA.

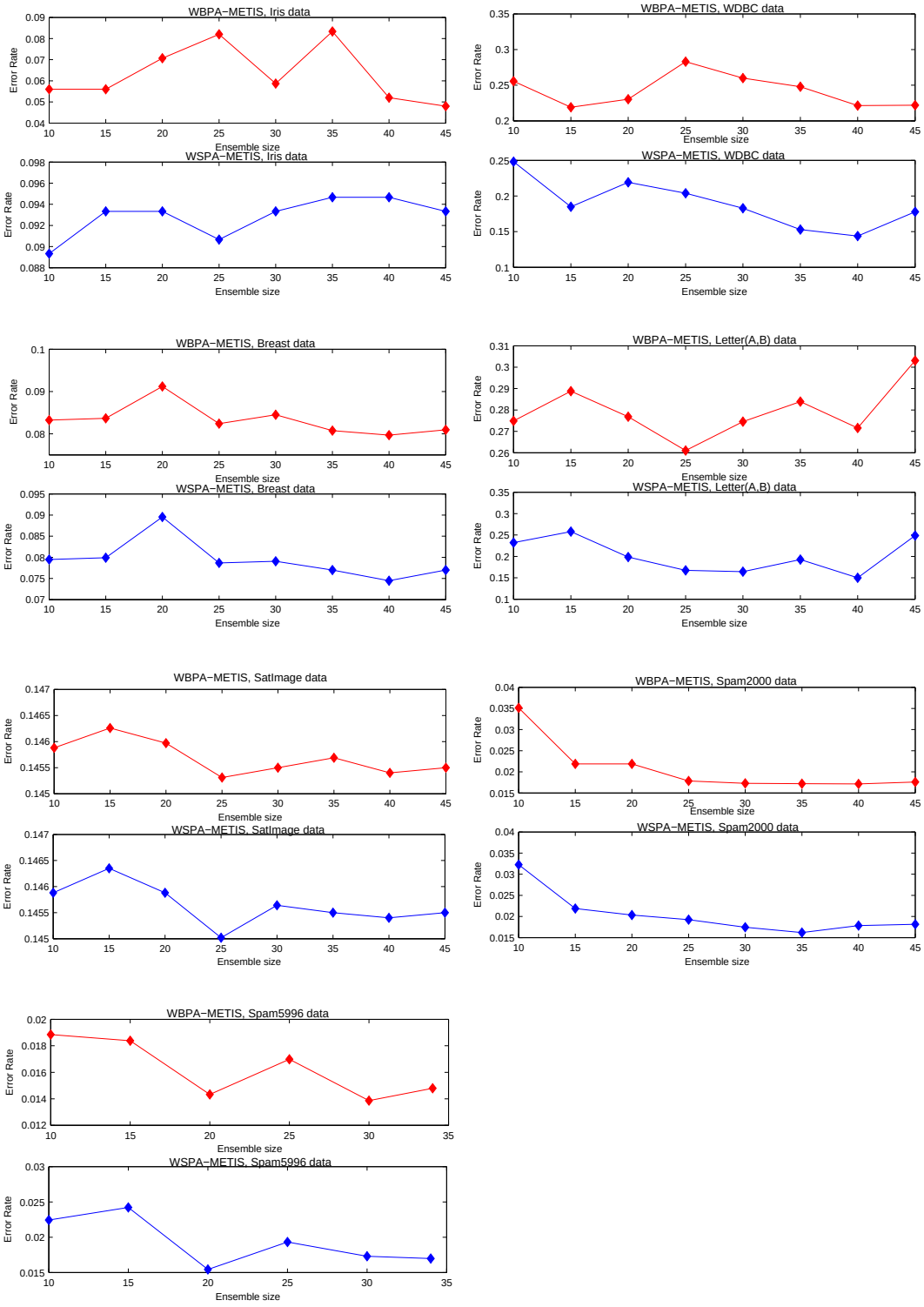

Figure 4.4: Error rate as a function of the ensemble size.

Table 4.9: Results on SatImage data

	Ens-NMI	Ens-ER	Max-NMI	Min-NMI	Avg-NMI	Max-ER	Min-ER	Avg-ER
WSPA-METIS	0.392	14.9	0.433	0.400	0.423	16.5	15.4	15.8
WSPA-SPEC	**0.467**	**13.2**	0.433	0.400	0.423	16.5	15.4	15.8
WBPA-METIS	0.389	15.0	0.433	0.400	0.423	16.5	15.4	15.8
WBPA-SPEC	**0.467**	**13.2**	0.433	0.400	0.423	16.5	15.4	15.8
WSBPA-METIS	0.416	14.5	0.433	0.400	0.423	16.5	15.4	15.8
WSBPA-SPEC	0.426	15.2	0.433	0.400	0.423	16.5	15.4	15.8
CSPA	0.273	20.3	0.433	0.400	0.423	16.5	15.4	15.8
MCLA	0.427	15.6	0.433	0.400	0.423	16.5	15.4	15.8
LAC+Co-as.	0.427	15.6	0.433	0.400	0.423	16.5	15.4	15.8
LAC+wCo-as.	0.428	16.1	0.433	0.400	0.423	16.5	15.4	15.8
k-means+Co-as.	0.424	15.7	0.426	0.423	0.424	15.7	15.6	15.7

Table 4.10: Results on Spam2000 data

	Ens-NMI	Ens-ER	Max-NMI	Min-NMI	Avg-NMI	Max-ER	Min-ER	Avg-ER
WSPA-METIS	0.939	0.7	0.945	0.889	0.923	1.5	0.6	0.9
WSPA-SPEC	0.894	1.4	0.945	0.889	0.923	1.5	0.6	0.9
WBPA-METIS	0.939	0.7	0.945	0.889	0.923	1.5	0.6	0.9
WBPA-SPEC	0.894	1.4	0.945	0.889	0.923	1.5	0.6	0.9
WSBPA-METIS	**0.946**	**0.6**	0.945	0.889	0.923	1.5	0.6	0.9
WSBPA-SPEC	**0.946**	**0.6**	0.945	0.889	0.923	1.5	0.6	0.9
CSPA	0.939	0.7	0.945	0.889	0.923	1.5	0.6	0.9
MCLA	0.940	0.7	0.945	0.889	0.923	1.5	0.6	0.9
LAC+Co-as.	0.940	0.7	0.945	0.889	0.923	1.5	0.6	0.9
LAC+wCo-as.	0.936	0.8	0.945	0.889	0.923	1.5	0.6	0.9
k-means+Co-as.	0.677	9.7	0.751	0.054	0.472	47.9	5.4	22.3

4.3 Diversity and Accuracy Analysis

4.3.1 Measures of Diversity and Accuracy

Diversity is an important aspect in building clustering ensembles. It is expected that the accuracy of the ensemble improves when a larger number of input clusterings is given, provided that the clusterings are diverse. Diversity in clustering ensembles is under investigation by many researchers. Here we study the interplay between accuracy and diversity for our ensemble techniques.

Table 4.11: Results on Spam5996 data

	Ens-NMI	Ens-ER	Max-NMI	Min-NMI	Avg-NMI	Max-ER	Min-ER	Avg-ER
WSPA-METIS	0.908	1.17	0.873	0.003	0.714	49.7	1.9	7.9
WSPA-SPEC	**0.933**	**0.93**	0.873	0.003	0.714	49.7	1.9	7.9
WBPA-METIS	0.908	1.12	0.873	0.003	0.714	49.7	1.9	7.9
WBPA-SPEC	**0.933**	**0.93**	0.873	0.003	0.714	49.7	1.9	7.9
WSBPA-METIS	**0.933**	**0.93**	0.873	0.003	0.714	49.7	1.9	7.9
WSBPA-SPEC	**0.933**	**0.93**	0.873	0.003	0.714	49.7	1.9	7.9
CSPA	0.898	1.3	0.873	0.003	0.714	49.7	1.9	7.9
MCLA	0.912	1.3	0.873	0.003	0.714	49.7	1.9	7.9
LAC+Co-as.	0.903	1.5	0.873	0.003	0.714	49.7	1.9	7.9
LAC-wCo-as.	0.899	1.6	0.873	0.003	0.714	49.7	1.9	7.9
k-means+Co-as.	0.749	5.4	0.749	0.006	0.39	49.7	5.4	41.2

[23] illustrate the importance of diversity for cluster ensemble accuracy. They measure diversity using NMI, a pairwise similarity measure that quantifies the information shared between two partitions, as defined in (4.1). Since NMI measures the similarity between two partitions, $(1 - NMI)$ gives the pairwise diversity. The pairwise measure of diversity, based on NMI, of an ensemble of L partitions is then defined as follows:

$$D_{NMI} = \frac{2}{L(L-1)} \sum_{i=1}^{L-1} \sum_{j=i+1}^{L} (1 - NMI(P_i, P_j)) \tag{4.2}$$

where P_i and P_j are two of the L partitions.

[23] confirm that high diversity in clustering results contributes to high quality in aggregation results. Moreover, their findings confirm that the quality of individual clusterings affects the aggregate ensembles. As expected, both a high level of diversity and the quality of individual clusterings are important in determining ensemble quality. The authors in [30] define a non-pairwise measure of diversity based on entropy; the larger the value of the entropy, the larger the diversity among partitions.

Entropy of a given collection of partitions is defined as follows:

$$H = \frac{2}{n(n-1)} \sum_{i=1}^{n} \sum_{j=i+1}^{n} -(p_{ij}\, log_2\, p_{ij} + (1 - p_{ij})\, log_2\, (1 - p_{ij})) \tag{4.3}$$

where n is the number of data points, and p_{ij} represents the proportion of times points $\mathbf{x}_i$ and $\mathbf{x}_j$ are clustered together. The authors discover that diversity among ensemble members is important in producing better ensemble results, but that diversity alone is not sufficient to guarantee good results. The selection of the ensemble method is also important. [48] and [32] discuss diversity and accuracy measures in great depth. In [48], the authors use three measures for diversity: Jaccard index, Adjusted Rand Index, and NMI. They find that diverse ensemble members produce better ensemble results than non-diverse members, even if non-diverse members are more accurate.

In particular, [32] investigate which diversity measure gives more accurate results. In all, six measures were examined. One is based on the Adjusted Rand Index (ARI), which measures the amount of departure from the assumption that any two clustering results have occurred by chance. ARI is a measure of similarity between two partitions, and is defined as follows:

$$t_1 = \sum_{i=1}^{c_A} \begin{pmatrix} n_i^A \\ 2 \end{pmatrix},\ t_2 = \sum_{j=1}^{c_B} \begin{pmatrix} n_j^B \\ 2 \end{pmatrix},\ t_3 = \frac{2t_1 t_2}{n(n-1)},$$

$$ar(A,B) = \frac{\sum_{i=1}^{c_A} \sum_{j=1}^{c_B} \begin{pmatrix} n_{ij} \\ 2 \end{pmatrix} - t_3}{\frac{1}{2}(t_1 + t_2) - t_3}, \tag{4.4}$$

where $\begin{pmatrix} a \\ b \end{pmatrix}$ is the binomial coefficient. A and B are two partitions of a dataset with n points, c_A and c_B are the number of clusters in partitions A and B respectively, n_i^A is the number of points in cluster i of partition A, n_j^B is the number of points in cluster j of partition B, and n_{ij} is the number of points cluster i of A and cluster j of B have in common. Since $ar()$ measures the similarity between two partitions, to compute the pairwise diversity one would consider $(1 - ar())$. Therefore, the measure of diversity, based on ARI, of an ensemble is defined as follows:

$$D_p = \frac{2}{L(L-1)} \sum_{i=1}^{L-1} \sum_{j=i+1}^{L} (1 - ar(P_i, P_j))$$

D_p measures the diversity of an ensemble with L partitions, where P_i, P_j are two such partitions.

Another measure discussed in [32] is entropy as defined in Equation (4.3) and discussed by Greene in [30]. Other measures evaluate individual (i.e, non-pairwise) diversities, by comparing individual clustering results with the ensemble result. One such measure is:

$$D_{np_1} = \frac{1}{L} \sum_{i=1}^{L} (1 - ar(P_i, P^*))$$

where P_i and P^* are the individual clustering result and the ensemble result respectively, and L is the number of clustering members.

An additional measure focuses on the spread of diversity (with respect to P^*) of

individual clusterings. It is defined as follows:

$$D_{np_2} = \sqrt{\frac{1}{L-1}\sum_{i=1}^{L}(1 - ar(P_i, P^*) - D_{np_1})^2}$$

Using this measure, [32] discover that a larger spread is not strongly related to the ensemble accuracy. To take this result into account, another measure was introduced:

$$D_{ARI} = \frac{1}{2}(1 - D_{np_1} + D_{np_2}) \tag{4.5}$$

which considers both variability and accuracy. Assuming that the ensemble result is close to the true labeling, we can measure the accuracy of individual clusterings by measuring how close they are to the ensemble result. Thus, a larger value of $(1 - D_{np_1})$ means higher accuracy. At the same time, variability within the ensemble can be measured using D_{np_2}. Equation (4.5) achieves a tradeoff between accuracy and variability.

The authors in [32] indicate that the most stable measures are D_{np_1} and D_{ARI}. The study focuses on the co-association approach to construct consensus functions. The authors conclude that an ensemble selected through medium diversity will fare better than either randomly selected ensembles or those selected through maximum diversity.

Based on the findings discussed above, we investigate here the issue of diversity and accuracy in more detail for our ensemble techniques (WSPA, WBPA, and WSBPA). Our objective is to investigate which measure of diversity is the best indicator for a good ensemble accuracy, and what is the preferred level of diversity (high, medium, or low). Such findings would enable one to select, from a set of ensembles, the one

that is most likely to provide good results. We consider two measures of diversity, one based on NMI as defined in (4.2), and one based on ARI as defined in (4.5). We observe that D_{NMI} is a pairwise diversity measure that does not depend on the ensemble methodology, while D_{ARI} is a non-pairwise diversity measure that depends on the ensemble methodology. Furthermore, we experiment with two methods to build a cluster ensemble: we run LAC with different values of h in one case, and with initial random centroids in the second case. In the following, we provide the details of the experiments, and discuss the results. The results obtained with random centroids are consistent with those obtained by varying h. Therefore, in the following, we omit the accuracy/diversity plots for random centroids.

4.3.2 Building cluster ensembles by varying h

To study how accuracy relates with the chosen measures of diversity, we created 50 ensembles of size 15 by varying the value of h. As clustering algorithm, we always used LAC. For each of the 50 ensembles, we computed both measures of diversity D_{NMI} and D_{ARI}, and corresponding accuracy values. In details, we ran the following procedure:

1. Run the LAC algorithm for $1/h = 1, \ldots, 50$;
2. Repeat the following 50 times:
   (a) Sample 15 clusterings out of the 50 generated in 1;
   (b) Run WBPA, WSPA, WSBPA (using METIS) on the selected 15 clusterings;
   (c) Compute the diversity measures D_{NMI} as in (4.2) and D_{ARI} as in (4.5), for $L = 15$;

(d) Compute the average accuracy of the ensemble components, both based on NMI and ARI, as follows:

$$Acc_{NMI} = \frac{1}{15}\sum_{i=1}^{15} NMI(P_i, P^T) \tag{4.6}$$

$$Acc_{ARI} = \frac{1}{15}\sum_{i=1}^{15} ar(P_i, P^T) \tag{4.7}$$

where P^T is the target partition (according to the ground truth);

(e) Compute the accuracy of the ensemble decision, both based on NMI and ARI, as follows:

$$Acc^*_{NMI} = NMI(P^*, P^T) \tag{4.8}$$

$$Acc^*_{ARI} = ar(P^*, P^T) \tag{4.9}$$

where P^* is the ensemble partition, and P^T is the target partition.

Figures 4.5-4.13 show the results of accuracy vs. diversity for our nine datasets. To construct the plots, we proceeded as follows. We sorted the 50 D_{NMI} values in increasing order. Each D_{NMI} value was associated with the corresponding Acc_{NMI} and Acc^*_{NMI} values. We plotted the collection of two dimensional points (D_{NMI}, Acc_{NMI}) and (D_{NMI}, Acc^*_{NMI}), and connected them with a line. We proceeded similarly for the measures based on ARI. This procedure was performed for each of the three ensemble techniques WSPA, WBPA, and WSBPA. In Figures 4.5-4.13, the points marked with a "$*$" symbol correspond to (D_{NMI}, Acc_{NMI}) and (D_{ARI}, Acc_{ARI}). The points marked with an "open square" symbol correspond to (D_{NMI}, Acc^*_{NMI}) and (D_{ARI}, Acc^*_{ARI}). From the plots, we observe the following:

1. Larger D_{NMI} (D_{ARI}) values give larger Acc_{NMI} (Acc_{ARI}) values, and larger Acc^*_{NMI} (Acc^*_{ARI}) values, for all datasets and all the three ensemble methods. This result suggests that, to obtain good ensemble accuracy, **a high level of diversity should be preferred**. (The same trend was obtained when diversity was generated by means of random centroids.)

2. For a given value of diversity D_{NMI} (D_{ARI}), the accuracy of the ensemble decision, Acc^*_{NMI} (Acc^*_{ARI}), is typically larger than the average accuracy of the ensemble components, Acc_{NMI} (Acc_{ARI}), for all three methods and all datasets (with few exceptions discussed below). **This demonstrates the efficacy of our ensemble methods**. Furthermore, the gain in accuracy, $Acc^*_{NMI} - Acc_{NMI}$ ($Acc^*_{ARI} - Acc_{ARI}$), in many cases is larger for larger diversity values (D_{NMI} and D_{ARI}, respectively). Again, **this confirms that a high level of diversity should be preferred**.

3. *WDBC dataset and WSBPA ensemble method*: For lower values of diversity (both based on NMI and ARI), the accuracy of the ensemble decision is very low, and slightly below the average accuracy of the ensemble components. As diversity increases, the ensemble accuracy improves rapidly, and achieves significant improvement upon the components. This case stresses the importance of high diversity. Note that, for this dataset, also the WSPA and WBPA techniques show a much larger accuracy gain for larger diversity values.

4. Results similar to WDBC are observed for the Letter(A,B) dataset, and accuracy/diversity measures based on NMI.

5. In general, given an ensemble of partitions, the average accuracy value of the components computed according to NMI (Acc_{NMI}) is higher than the average accuracy value computed according to ARI (Acc_{ARI}). This is due to the fact that

$NMI() \in [0, 1]$, while $ar() \in [-1, 1]$. Thus, the summation in (4.7) may contain negative values, which lead to smaller averages than in (4.6) (where the smallest components are zeros). On the other hand, the values Acc^*_{NMI} and Acc^*_{ARI}, which measure the accuracy of the ensemble partitions, are in general closer to each other. This happens because the largest value both for $NMI()$ and $ar()$, in (4.8) and (4.9) respectively, is 1. This different scaling of the accuracy/diversity measures causes the values (D_{NMI}, Acc^*_{NMI}) to lie below the (D_{NMI}, Acc_{NMI}) values for the SatImage dataset (while the opposite trend is observed for the measures based on ARI) (see Figure 4.11). We also observe that the range for the diversity values is very narrow in this case, suggesting the presence of correlated partitions in the ensembles. According to Table 4.9, our three ensemble techniques provide a smaller error rate than the minimum error rate of the input clusterings. **This suggests that a measure of accuracy/diversity based on ARI might be more robust and consistent than a measure based on NMI**. The results obtained in [32] corroborate our conclusions. In fact, [32] also indicate that the most stable measures of diversity are based on ARI. Furthermore, their study focuses on the co-association approach to construct consensus functions. This provides evidence that D_{ARI} is a good measure of diversity for both co-association based and graph partitioning based ensemble scenarios. This is an interesting and relevant result since D_{ARI} *does depend* on the ensemble methodology.

4.3.3 Ensemble Selection Method: Medium vs. High Diversity

To verify whether the ensemble selection method based on diversity is effective, and which measure of diversity is a better index, we performed the following experiment:

1. Construct 50 ensembles of size 15 (e.g., using random centroids or varying h);
2. Compute the average accuracy $Acc_{NMI} = \frac{1}{50}\sum_{j=1}^{50} NMI(P_j^*, P^T)$;
3. Randomly select 15 ensembles out of the 50;
4. Compute D_{NMI} for the 15 ensembles; order the D_{NMI} values, and identify the ensembles with median and maximum diversity. Compute their accuracy: $NMI(P_{med}^*, P^T)$ and $NMI(P_{max}^*, P^T)$;
5. Repeat steps 3 and 4 50 times;
6. Compute the average accuracy of the ensembles with median (maximum) diversity:

$$Acc_{NMI}^{med} = \frac{1}{50}\sum_{i=1}^{50} NMI(P_{i,med}^*, P^T)$$

$$Acc_{NMI}^{max} = \frac{1}{50}\sum_{i=1}^{50} NMI(P_{i,max}^*, P^T)$$

The above steps were executed also using ARI. In Tables 4.12-4.14 we report the results obtained for WSPA, WBPA, and WSBPA respectively, varying the parameter h over the values $\{1, 2, \ldots, 50\}$. Similar results were obtained by using random centroids. For WSPA and WBPA, the selection procedure based on Acc_{ARI}^{max} provides the highest accuracy value averaged across all tested datasets. In particular, for WSPA,

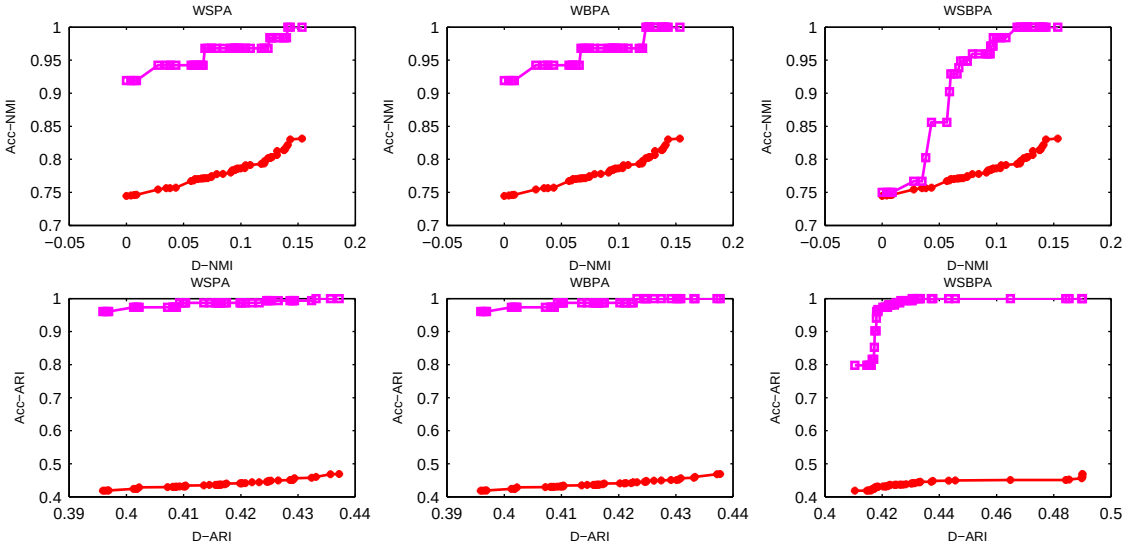

Figure 4.5: Two Gaussian dataset: Accuracy vs. Diversity

$\overline{Acc_{ARI}^{max}} = 0.81$ vs. $\overline{Acc_{ARI}} = 0.76$, and $Acc_{ARI}^{max} \geq Acc_{ARI}$ for all datasets. For WBPA, $\overline{Acc_{ARI}^{max}} = 0.78$ vs. $\overline{Acc_{ARI}} = 0.74$, and $Acc_{ARI}^{max} \geq Acc_{ARI}$ for all datasets but one (Iris). For WSBPA, the selection procedure based on Acc_{ARI}^{med} provides the highest accuracy value averaged across the datasets. In this case, though, the improvement over the entire collection of ensembles is less significant ($\overline{Acc_{ARI}^{med}} = 0.67$ vs. $\overline{Acc_{ARI}} = 0.66$).

For all three methods, the ensemble selection method using the non-pairwise diversity measure D_{ARI} appears to be the most effective. This finding is consistent with our previous observations. Furthermore, for the WSPA and WBPA methods, a high level of diversity is preferred, while for WSBPA a medium level of diversity appears to be better on average. Due to the dependency on the specific dataset, though, results are less significant and conclusive for WSBPA.

Table 4.12: Ensemble accuracy: medium vs. high diversity (WSPA)

	Acc_{NMI}	Acc_{NMI}^{max}	Acc_{NMI}^{med}	Acc_{ARI}	Acc_{ARI}^{max}	Acc_{ARI}^{med}
Two-Gaussian	0.96	**0.99**	0.97	0.98	0.99	0.99
Three-Gaussian	0.99	0.99	1.0	0.99	0.99	1.0
Iris	0.75	0.74	0.75	0.75	0.75	0.76
Breast	0.62	0.72	0.63	0.72	0.82	0.72
Letter (A,B)	0.36	0.58	0.41	0.44	0.66	0.57
SatImage	0.40	0.40	0.40	0.50	0.50	0.50
Spam2000	0.86	0.85	0.86	0.92	0.93	0.93
Averages	0.71	0.75	0.72	0.76	**0.81**	0.78

Table 4.13: Ensemble accuracy: medium vs. high diversity (WBPA)

	Acc_{NMI}	Acc_{NMI}^{max}	Acc_{NMI}^{med}	Acc_{ARI}	Acc_{ARI}^{max}	Acc_{ARI}^{med}
Two-Gaussian	0.97	0.99	0.97	0.98	0.99	0.99
Three-Gaussian	0.99	0.98	0.99	0.99	0.99	0.99
Iris	0.83	0.80	0.83	0.85	0.81	0.86
Breast	0.60	0.69	0.61	0.71	0.80	0.71
Letter (A,B)	0.17	0.24	0.18	0.23	0.46	0.17
SatImage	0.40	0.40	0.40	0.50	0.50	0.50
Spam2000	0.85	0.77	0.86	0.91	0.94	0.92
Averages	0.69	0.70	0.69	0.74	**0.78**	0.73

Table 4.14: Ensemble accuracy: medium vs. high diversity (WSBPA)

	Acc_{NMI}	Acc_{NMI}^{max}	Acc_{NMI}^{med}	Acc_{ARI}	Acc_{ARI}^{max}	Acc_{ARI}^{med}
Two-Gaussian	0.94	1.0	0.96	0.96	0.82	0.99
Three-Gaussian	0.94	0.94	0.94	0.96	0.97	0.96
Iris	0.77	0.78	0.77	0.80	0.79	0.80
Breast	0.27	0.40	0.25	0.20	0.11	0.18
Letter (A,B)	0.24	0.35	0.27	0.29	0.40	0.31
SatImage	0.41	0.42	0.41	0.50	0.50	0.50
Spam2000	0.83	0.73	0.85	0.90	0.93	0.92
Averages	0.63	0.66	0.64	0.66	0.65	**0.67**

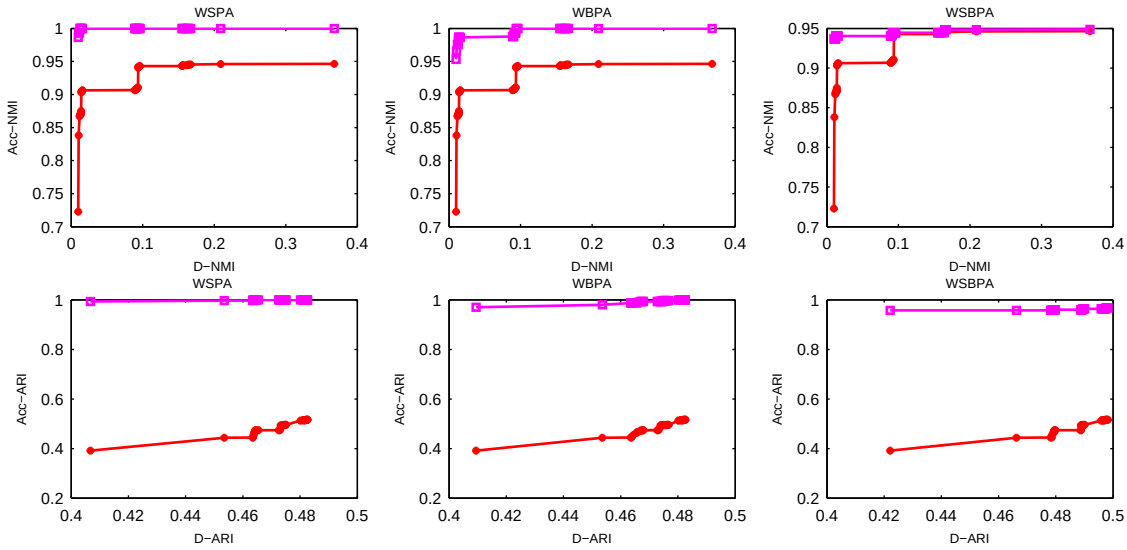

Figure 4.6: Three Gaussian dataset: Accuracy vs. Diversity

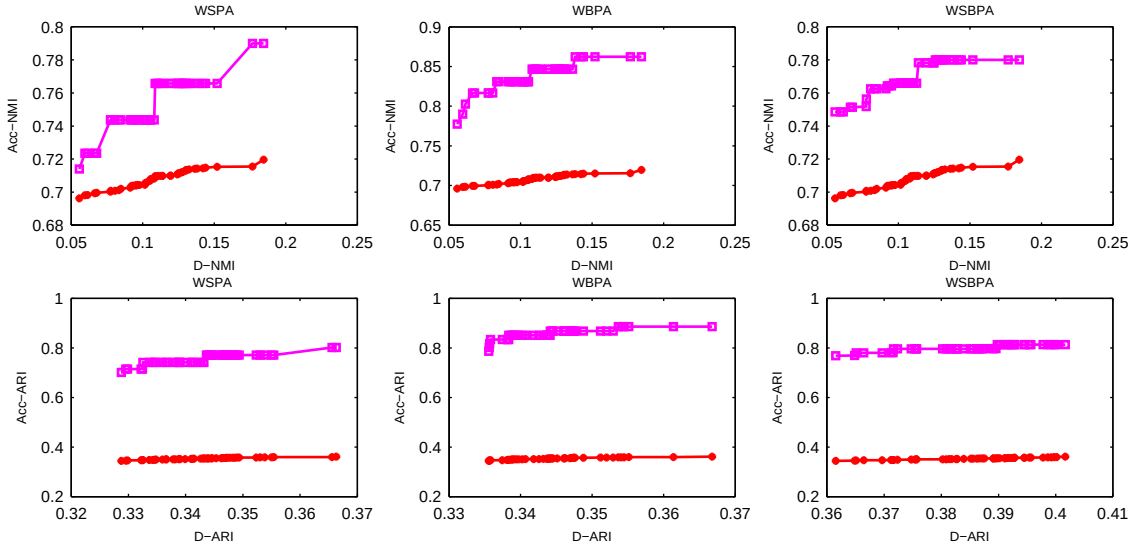

Figure 4.7: Iris dataset: Accuracy vs. Diversity

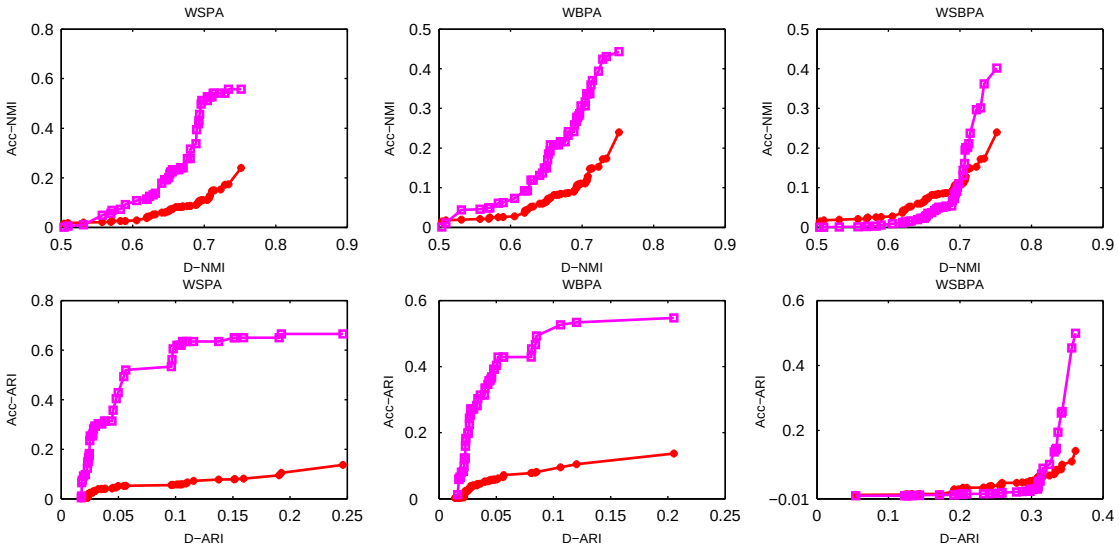

Figure 4.8: WDBC dataset: Accuracy vs. Diversity

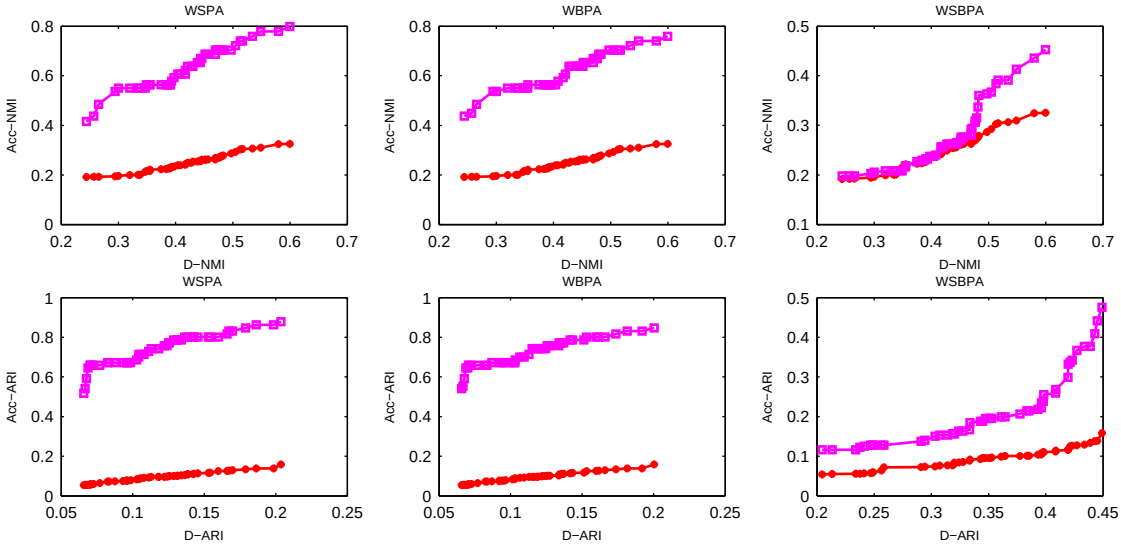

Figure 4.9: Breast dataset: Accuracy vs. Diversity

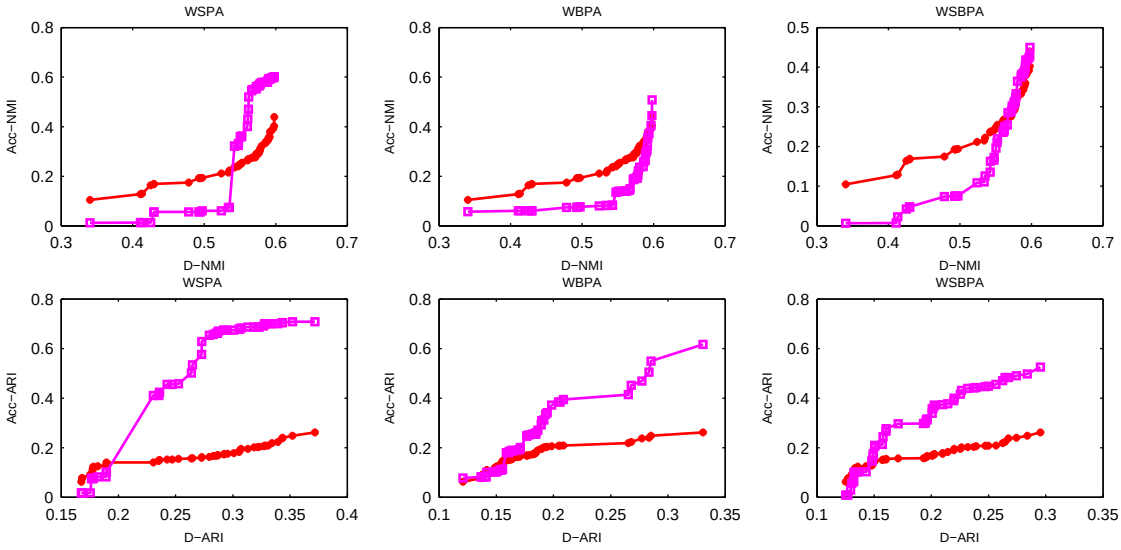

Figure 4.10: Letter(A,B) dataset: Accuracy vs. Diversity

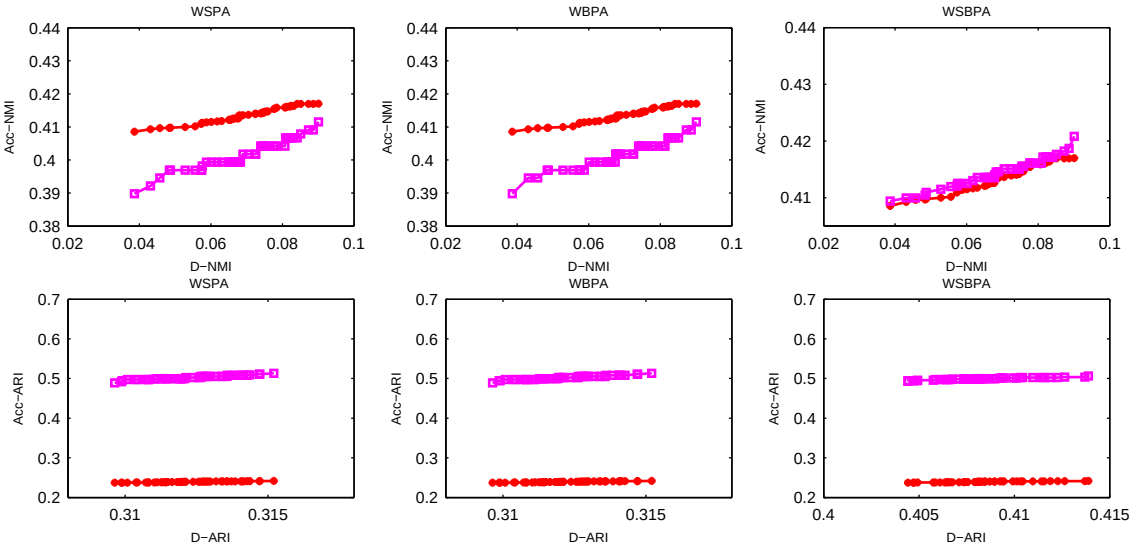

Figure 4.11: SatImage dataset: Accuracy vs. Diversity

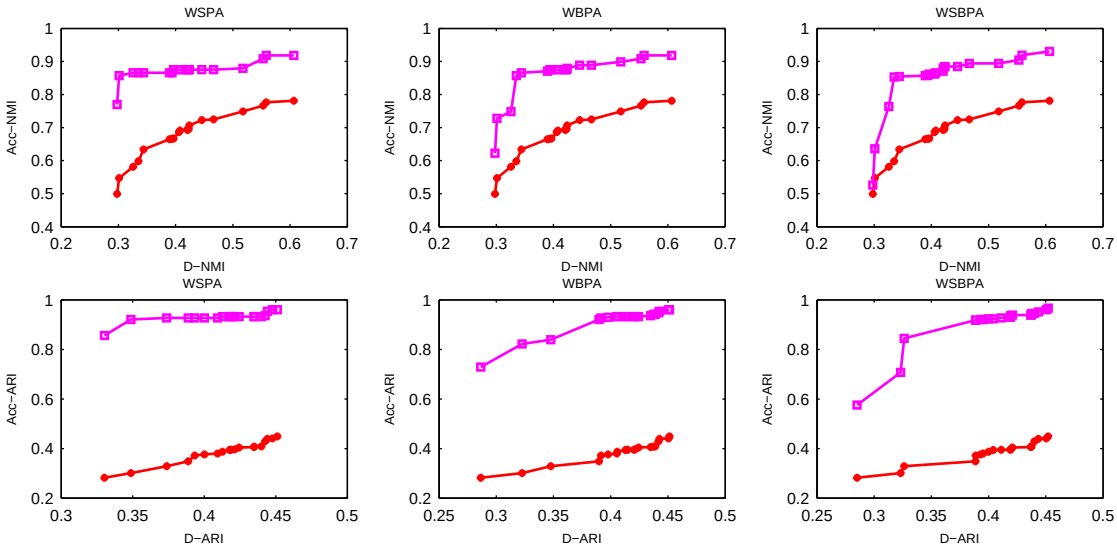

Figure 4.12: Spam2000 dataset: Accuracy vs. Diversity

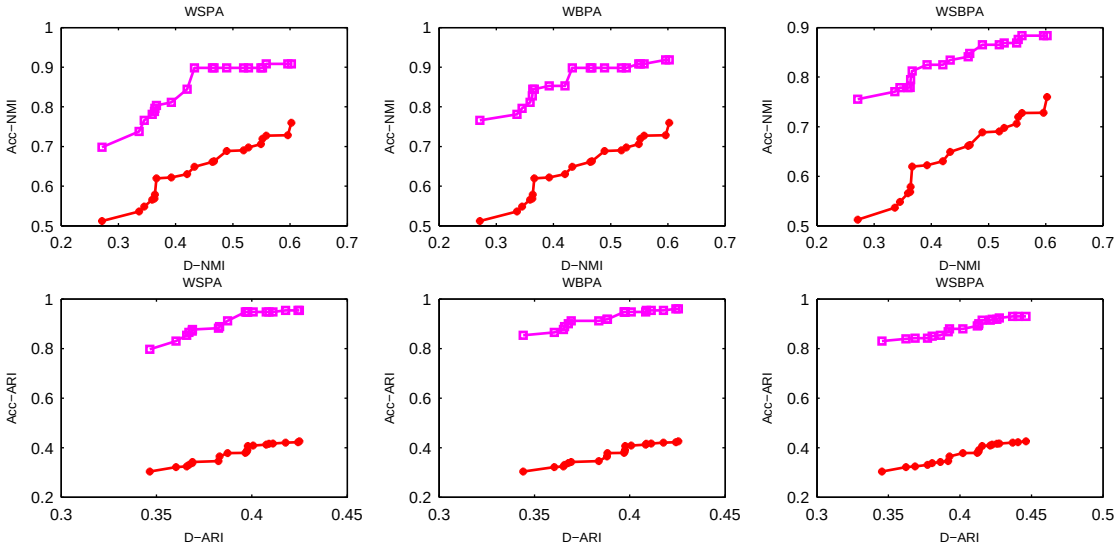

Figure 4.13: Spam5996 dataset: Accuracy vs. Diversity

Chapter 5: Applications

In the previous chapters we have introduced our ensemble techniques and demonstrated empirically their efficacy. Here we discuss the usage of our ensembles techniques in two different applications. First, we investigate the feasibility of our subspace ensemble technique (WSBPA) for the categorization of unlabeled documents. Second, we generalize the usage of our consensus functions WSPA and WBPA with categorical data.

5.1 Categorization of Unlabeled Documents

We investigate the use of our subspace cluster ensemble technique (WSBPA) for the categorization of unlabeled documents. The output of WSBPA is twofold: it provides a partition of the data and a measure of local feature relevance for each identified group of data. For text documents, the analysis of relevance values (i.e., weights) credited to features (i.e., terms) can assist the identification of descriptive words representative of topics discussed in the documents.

To demonstrate these concepts we performed experiments with two datasets: spam Email-1431 and 20 Newsgroups. To reduce the dimensionality of the data, we followed the procedure presented in [42]. Documents were first preprocessed by eliminating stop and rare words, and by stemming words to their root source. A global unsupervised feature selection procedure, based on frequent itemset mining, was then applied. The objective of this step is to identify sets of terms that co-occur frequently

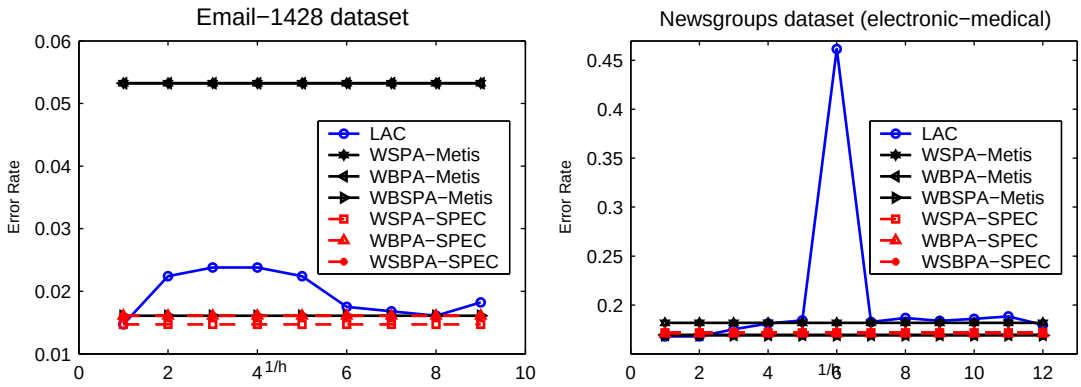

Figure 5.1: Results on text datasets. (Left): Email-1431 dataset. (Right): 20 Newsgroups dataset (electronic-medical)

in the given corpus of documents. Such terms become the features used in the final representation of documents.

Email-1431 is the same dataset used in the experiments described in Section 4.1. The original size of the dictionary is 38,713. After processing the data as described above, the dictionary size was reduced to 285. As before, we ran a two-class classification problem by merging the conference and jobs emails into one group (non-spam). 20 Newsgroups is a collection of 20,000 messages collected from 20 different netnews newsgroups. One thousand messages from each of the 20 newsgroups were chosen at random and partitioned by newsgroups name. In our experiments we consider the categories medical (990) and electronics (981). The original size of the dictionary is 24,546; after processing the data, the dictionary size was reduced to 321.

Tables 5.1 and 5.2 report the results we obtained for these two datasets. We ran our three methods (WSPA, WBPA, and WSBPA) using both METIS and spectral clustering. We report the ensemble NMI, the ensemble error rate, and minimum, maximum and average error rates of the input clusterings. (Figure 5.1 shows the ranges of values for the parameter h used to construct the ensembles.)

WSBPA gives good results in both cases. We observe that for the Email-1431

dataset, WSBPA gives the same error rate (1.6%) when combined with either METIS or spectral clustering (as shown in Figure 5.1 (Left) and in Table 5.1). Such error rate is very close to the minimum error rate provided by the runs of LAC. Moreover, WSBPA significantly outperforms WSPA and WBPA when METIS is used. With SPEC, all three methods provide similar results. The fact that SPEC performs better than METIS might be due to the slightly unbalanced data (786 spams vs. 642 non-spams). Also for the 20 Newsgroups dataset (electronic, medical), WSBPA gives an error rate that is very close to the minimum error rate provided by LAC (for both METIS and SPEC) (see Figure 5.1 (Right) and Table 5.2). In this case, METIS and SPEC give similar results (the dataset is balanced).

5.1.1 Analysis of Weights

We analyzed the weights credited to features by the algorithm WSBPA (combined with METIS). The analysis of weights assigned to words provides some insights on the nature of the spam filtering problem and the general classification case. As Figures 5.2 and 5.3 show, the selected words (i.e., those words that receive largest weight values) are representative of the underlying categories, which provides evidence that our subspace cluster ensemble technique is capable of sifting relevant words, while discarding (i.e., assigning a low weight value) spurious ones.

Let us consider the distribution of weights obtained for the Email-1431 dataset. Figure 5.2 shows the weight values and corresponding words for the two class case (the non-spam class corresponds to both conference and jobs emails). Here we plot the top words that received highest weight for each class (discarding those without a clear meaning, e.g., abbreviations, acronyms, etc.). We observe that words reflecting the topic of a category receive a larger weight in the *other class*. For example, the

words "*sales*", "*money*", "*marketing*", "*credit*", etc. get a larger weight in the non-spam category (their weights in the spam class are very close to zero). Similarly, the words "*computational*", "*neuroscience*", "*neural*", "*algorithms*", "*deadline*", etc. receive larger weights in the spam category. The weights for these words in the non-spam class are very close to zero. While surprising at first, this trend may be due to the nature of the spam and non-spam email distributions. Each of these two categories is actually a combination of subclasses. The non-spam class in this case is the union of conference and jobs emails (by construction). Likewise, the spam messages can be very different in nature (sales, jokes, diets, fraud, etc.), and therefore different in their word content. As a consequence, the variance of feature values for words reflecting the general topic of a category is larger within the same category than in the other one (e.g., the word "*sales*" appears only in half of the spam messages, and does not appear in any of the non-spam emails). Since the weights computed by the LAC algorithm are inversely proportional to a measure of such variance of values (i.e., X_{js}), we obtain the "swapping phenomenon" depicted in Figure 5.2. This analysis can be interpreted as the fact that the absence of a certain term (e.g., absence of the word "*sales*" within the non-spam messages) is a characteristic shared across the emails of a given category; whereas the presence of certain words shows a larger variability across emails of a given category (e.g., the word "*sales*" appears only in half of the spam messages).

Figure 5.3 shows the weight values and corresponding words for the 20 Newsgroups (electronic, medical) dataset. In this case words receive largest weights within the representative class (e.g., "*system*", "*noise*", "*circuit*", "*range*", for the electronic class; "*screen*", "*hot*", "*dead*", "*cost*", "*program*" for the medical class). In this case, categories represent focused topics, and therefore words reflecting the content of documents show a small variance (e.g., the word "*system*" appears in all documents on

electronics, and thus its variance is zero).

For this dataset, we also analyzed the dictionary of the corpus, and noticed that the majority of words is descriptive of the electronic category, while the medical domain is under represented. This bias was also reflected within the words that received larger weights: we could easily identify many words of the electronic domain, while words from the medical domains were less in number. Given the biased dictionary, this result is expected.

The above results provide evidence that the weights computed by the WSBPA algorithm are meaningful, that is the averaging of weights performed by Equation (3.7) properly captures the local relevance of features. This is important for the cluster prediction of future data. Local weights also provide information regarding the subspace each cluster belongs to, thus allowing data interpretation, and possibly data compression. Specifically, for text categorization, the analysis of weights can be informative of the nature of the categorization problem, and can be used to guide the process of text interpretation. Of course, we are not advocating that local weights alone can solve the problem of automatic document annotation. Our results simply show that they are useful for the identification of descriptive words. Local weights alone, though, are not able to account for all possible configurations and words' distributions. For example, a word that appears in all documents of one class and in zero documents of the other, receives large weight in both (its variance is zero in both cases). Considering the frequency of occurrence within each class, may clarify which class the word is descriptive of. While this phenomenon was not observed in our data, one has to account for such instances in general. Considering relative frequencies of words that receive large weight in both classes is a viable solution.

Table 5.1: Results on Email-1431

	Ens-NMI	Ens-ER	Min-ER	Max-ER	Avg-ER
WSPA-METIS	0.741	5.3	1.5	2.2	1.95
WBPA-METIS	0.741	5.3	1.5	2.2	1.95
WSBPA-METIS	0.880	1.6	1.5	2.2	1.95
WSPA-SPEC	0.889	1.5	1.5	2.2	1.95
WBPA-SPEC	0.880	1.6	1.5	2.2	1.95
WSBPA-SPEC	0.880	1.6	1.5	2.2	1.95

Table 5.2: Results on 20 Newsgroups (electronic, medical)

	Ens-NMI	Ens-ER	Min-ER	Max-ER	Avg-ER
WSPA-METIS	0.316	18.16	16.79	46.17	20.37
WBPA-METIS	0.344	16.95	16.79	46.17	20.37
WSBPA-METIS	0.353	16.89	16.79	46.17	20.37
WSPA-SPEC	0.343	17.15	16.79	46.17	20.37
WBPA-SPEC	0.344	17.09	16.79	46.17	20.37
WSBPA-SPEC	0.345	17.19	16.79	46.17	20.37

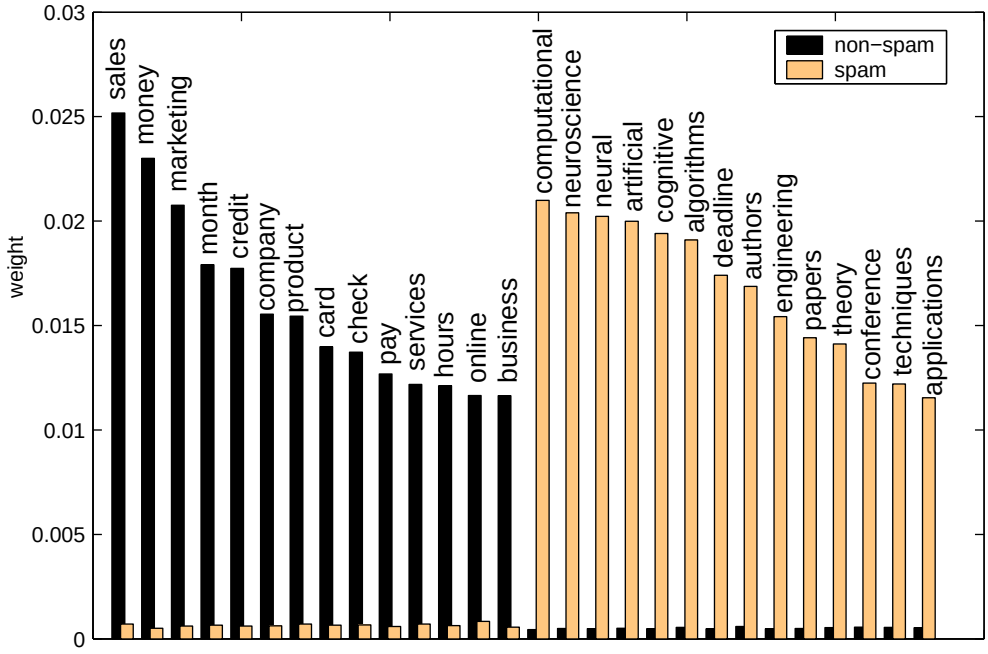

Figure 5.2: Email-1431: Words and corresponding weight values.

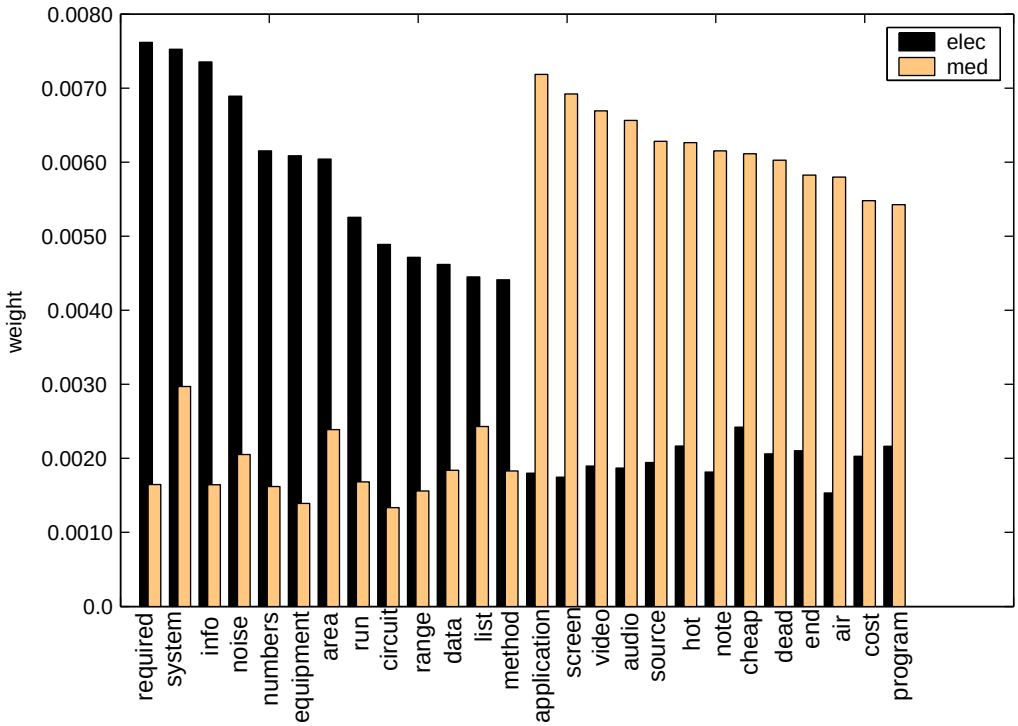

Figure 5.3: 20 Newsgroups (electronic, medical): Words and corresponding weight values.

5.2 Clustering Ensembles for Categorical Data

5.2.1 Introduction

Clustering ensembles have been successfully designed for numerical features. On the other hand, clustering ensembles for categorical data have not received much attention in the literature. Categorical data present its own array of difficulties. First, because their relationship is not linear, categorical attributes have no single ordering. Second, because categorical attributes do not have the inherent geometrical structure implied by numerical values, traditional distance and similarity measures will not yield accurate results.

Although these difficulties are steep, they are not insurmountable. Clustering techniques have been successfully applied to categorical data. However, these methods have limitations similar to those found in numerical clusters. Difficulties include the question of how to tune input parameters without a prior knowledge and the inaccuracy that result in high dimensions.

In our work, we attempted to overcome the faults inherent to categorical clustering methods by introducing two consensus functions for categorical data. Our techniques use the clustering results produced by COOLCAT, but our methods are flexible enough to be used in combination with any categorical clustering approach.

Our categorical clustering ensemble applications contribute to a currently underserved area in the literature. Our applications filter out spurious structures identified by individual runs of the clustering algorithm, and achieve results with a high level of accuracy. Our methods surpasses previous categorical clustering ensembles in their ability to handle data in high dimensions as will be described next.

5.2.2 Enhancement of COOLCAT

To enhance the accuracy of the clustering given by COOLCAT, we construct an ensemble of clusterings obtained by multiple runs of the algorithm. A good accuracy-diversity trade-off must be achieved to obtain a consensus solution that is superior to the components. To improve the diversity among the ensemble components, each run of COOLCAT operates within a random subspace of the feature space, obtained by random sampling a fixed number of attributes from the set of given ones. Thus, diversity is guaranteed by providing the components different *views* (or projections) of the data. Since such views are generated randomly from a (typically) large pool of attributes, it is highly likely that each component receives a different prospective of the data, which leads to the discovery of diverse (and complementary) structures within the data.

The rationale behind our approach finds its justification in classifier ensembles, and in the theory of *Stochastic Discrimination* [44, 45]. The advantages of a random subspace method, in fact, are well known in the context of ensembles of classifiers [37, 60].

Furthermore, we observe that performing clustering in random subspaces should be advantageous when data present redundant features, and/or the discrimination power is spread over many features, which is often the case in real life scenarios. Under these conditions, in fact, redundant/noisy features are less likely to appear in random subspaces. Moreover, since discriminant information is distributed across several features, we can generate multiple meaningful (for cluster discrimination) subspaces. This is in agreement with the assumptions made by the theory of stochastic discrimination [45] for building effective ensembles of classifiers; that is, there exist multiple sets of features able to discern between training data in different classes, and unable to discern training and testing data in the same class.

5.2.3 Categorical Similarity Partitioning Algorithm (CSPA)

Our aim is to generate robust and stable solutions via a consensus clustering method. We can generate contributing clusterings by running multiple times the COOLCAT algorithm within random subspaces. Thus, each ensemble component has access to a random sample of f features drawn from the original d dimensional feature space. The objective is then to find a consensus partition from the output partitions of the contributing clusterings, so that an "improved" overall clustering of the data is obtained.

In order to derive our consensus function, for each data point $\mathbf{x}_i$ and each cluster C_l, we want to define the probability associated with cluster C_l given that we have observed $\mathbf{x}_i$. Such probability value must conform to the information provided by a given component clustering of the ensemble. The consensus function will then aggregate the findings of each clustering component utilizing such probabilities.

COOLCAT partitions the data into k distinct clusters. In order to compute distances between data points and clusters, we represent clusters using *modes*. The mode of a cluster is the vector of the most frequent attribute values in the given cluster. In particular, when different values for an attribute have the same frequency of occurrence, we consider the whole data set, and choose the attribute that has the least overall frequency. Ties are broken randomly.

We then compute the distance between a point $\mathbf{x}_i$ and a cluster C_l by considering the *Jaccard distance* [53] between $\mathbf{x}_i$ and the mode $\mathbf{c}_l$ of cluster C_l, defined as follows:

$$d_{il} = 1 - \frac{|\mathbf{x}_i \cap \mathbf{c}_l|}{|\mathbf{x}_i \cup \mathbf{c}_l|} \tag{5.1}$$

where $|\mathbf{x}_i \cap \mathbf{c}_l|$ represents the number of matching attribute values in the two vectors,

and $|\mathbf{x}_i \cup \mathbf{c}_l|$ is the number of distinct attribute values in the two vectors.

We then follow the same steps as in the Weighted Similarity Partitioning algorithm (WSPA) presented in Section 3.2.1

5.2.4 Categorical Bipartite Partitioning Algorithm (CBPA)

Our second approach (CBPA) maps the problem of finding a consensus partition to a bipartite graph partitioning problem. We follow the same steps as in the Weighted Bipartite Partitioning algorithm (WBPA) presented in Section 3.2.2, however we calculate the distances between a point $\mathbf{x}_i$ and a cluster C_l by considering the *Jaccard distance* [53] as explained in the previous section.

5.2.5 Experimental Design and Results

In our experiments, we used four real datasets. The characteristics of all datasets are given in Table 5.3. The Archeological dataset is taken from [5], and was used in [12] as well. Soybeans, Breast, and Congressional Votes are from the UCI Machine Learning Repository [8].

The Soybeans dataset consists of 47 samples and 35 attributes. Since some attributes have only one value, we have removed them, and selected the remaining 21 attributes for our experiments, as it has been done in other research [27]. For the Breast-cancer data, we sub-sampled the most populated class from 444 to 239 as we have conducted in our previous work to obtain balanced data [4]. The Congressional Votes dataset contains attributes which consist of either 'yes' or 'no' responses; we treat missing values as an additional domain attribute value for each feature as conducted in [12].

Evaluating the quality of clustering is in general a difficult task. Since class labels

Table 5.3: Characteristics of the data

Dataset	k	D	n (points-per-class)
Archeological	2	8	20 (11-9)
Soybeans	4	21	47 (10-10-10-17)
Breast-cancer	2	9	478 (239-239)
Vote	2	16	435 (267-168)

are available for the datasets used here, we evaluate the results by computing the error rate and the normalized mutual information (NMI) 4.1 described in Section (4.1).

5.2.6 Analysis of the Results

For each dataset, we ran COOLCAT 10 times with different sets of random features. The number f of selected features was set to half the original dimensionality for each data set: $f = 4$ for Archeological, $f = 11$ for Soybeans, $f = 5$ for Breast-cancer, and $f = 8$ for Vote. The clustering results of COOLCAT are then given as input to the consensus clustering techniques being compared. (As value of k, we input both COOLCAT and the ensemble algorithms the actual number of classes in the data.)

Figures 5.4-5.7 plot the error rate (%) achieved by COOLCAT in each random subspace, and the error rates of our categorical clustering ensemble methods (CSPA-Metis, CSPA-SPEC, CBPA-Metis, and CBPA-SPEC, where SPEC is short for spectral clustering). We also plot the error rate achieved by COOLCAT over multiple runs in the entire feature space. The figures show that we were able to obtain diverse clusterings within the random subspaces. Furthermore, the instable performance of COOLCAT in the original space shows its sensitivity to the initial random seeding process, and to the order according to which data are processed. (We kept the input parameters of COOLCAT fixed in all runs: the sample size was set to 8, and the

reprocessing size was set to 10.)

Detailed results for all data are provided in Tables 5.4-5.7, where we report the NMI and error rate (ER) of the ensembles, as well as the maximum, minimum, and average NMI and error rate values for the input clusterings.

In general, our ensemble techniques were able to filter out spurious structures identified by individual runs of COOLCAT, and performed quite well. Our techniques produced error rates comparable with, and sometime better than, COOLCAT's minimum error rate. CSPA-Metis provided the lowest error rate among the methods being compared on three data sets. For the Archeological and Breast-cancer data, the error rate provided by the CSPA-Metis technique is as good or better than the best individual input clustering. It is worth noticing that for these two datasets, CSPA-Metis gave an error rate which is lower than the best individual input clustering on the entire feature space (see Figures 5.4 and 5.6) In particular, on the Breast-cancer data all ensemble techniques provided excellent results. For the Soybeans dataset, the error rate of CSPA-Metis is still well below the average of the input clusterings, and for Vote is very close to the average.

Also CBPA (both with Metis and SPEC) performed quite well. In general, it produced error rates comparable with the other techniques. CBPA produced error rates well below the average error rates of the input clusterings, with the exception of the Vote dataset. For the Vote data, all ensemble methods gave error rates close to the average error rate of the input clusterings. In this case, COOLCAT on the full space gave a better performance.

Overall, our categorical clustering ensemble techniques are capable of boosting the performance of COOLCAT, and achieve more robust results. Given the competitive behavior previously shown by COOLCAT, the improvement obtained by our ensemble techniques is a valuable achievement.

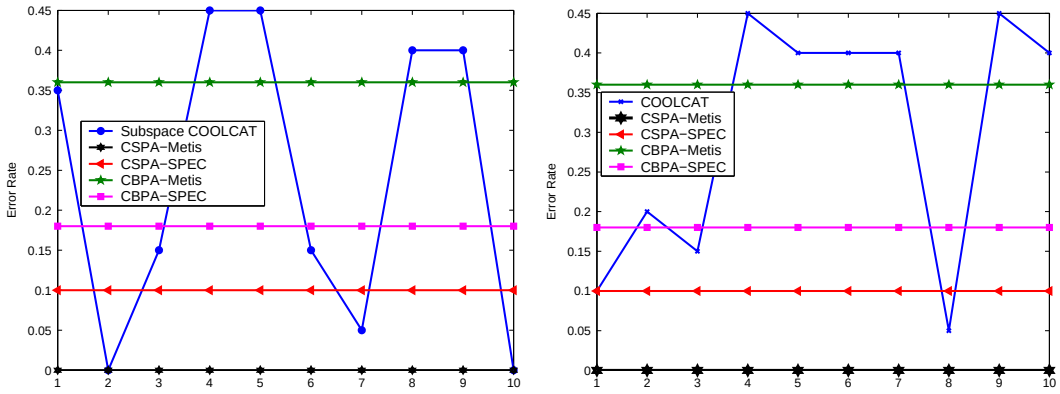

Figure 5.4: (Left): Archeological data: error rates of cluster ensemble methods, and COOLCAT in random subspaces. (Right):Archeological data: error rates of cluster ensemble methods, and COOLCAT using all features.

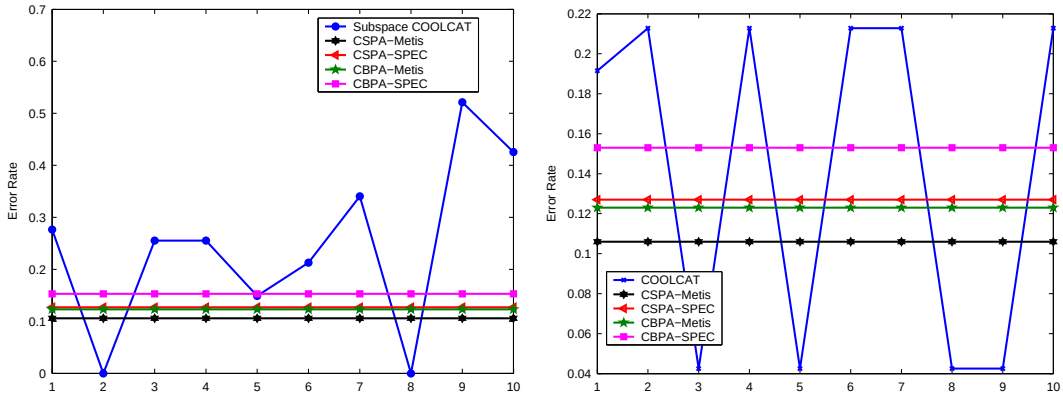

Figure 5.5: (Left): Soybeans data: error rates of cluster ensemble methods, and COOLCAT in random subspaces. (Right):Soybeans data: error rates of cluster ensemble methods, and COOLCAT using all features.

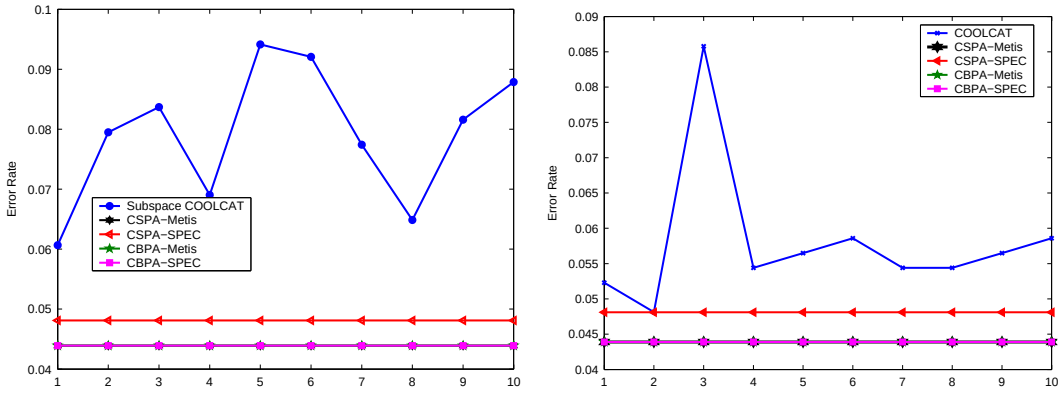

Figure 5.6: (Left): Breast-cancer data: error rates of cluster ensemble methods, and COOLCAT in random subspaces. (Right): Breast-cancer data: error rates of cluster ensemble methods, and COOLCAT using all features.

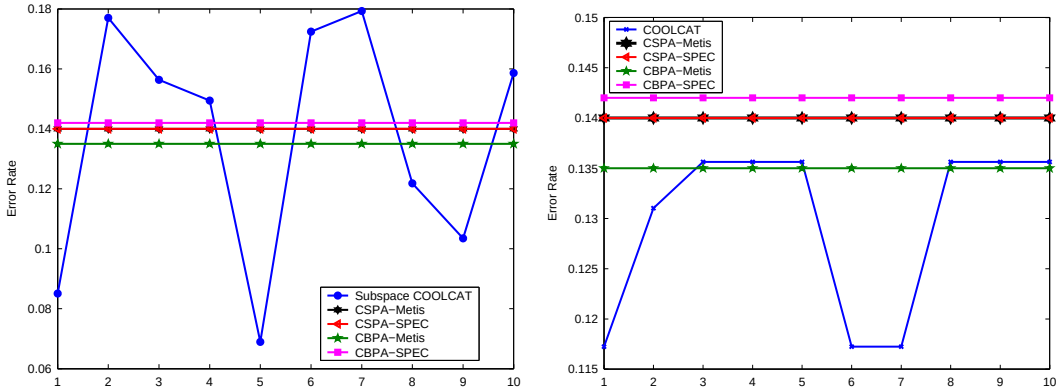

Figure 5.7: (Left): Vote data: error rates of cluster ensemble methods, and COOLCAT in random subspaces. (Right): Vote data: error rates of cluster ensemble methods, and COOLCAT using all features.

Table 5.4: Results on Archeological data

	Ens-NMI	Ens-ER	Max-ER	Min-ER	Avg-ER	Max-NMI	Min-NMI	Avg-NMI
CSPA-METIS	**1**	**0**	45.00	0	24.00	1	0.033	0.398
CSPA-SPEC	0.21	36.00	45.00	0	24.00	1	0.033	0.398
CBPA-METIS	0.5284	10.00	45.00	0	24.00	1	0.033	0.398
CBPA-SPEC	0.603	18.0	45.00	0	24.00	1	0.033	0.398

Table 5.5: Results on Soybeans data

	Ens-NMI	Ens-ER	Max-ER	Min-ER	Avg-ER	Max-NMI	Min-NMI	Avg-NMI
CSPA-METIS	0.807	**10.6**	52.1	0	24.4	1	0.453	0.689
CSPA-SPEC	0.801	12.3	52.1	0	24.4	1	0.453	0.689
CBPA-METIS	0.761	12.8	52.1	0	24.4	1	0.453	0.689
CBPA-SPEC	0.771	15.3	52.1	0	24.4	1	0.453	0.689

Table 5.6: Results on Breast cancer data

	Ens-NMI	Ens-ER	Max-ER	Min-ER	Avg-ER	Max-NMI	Min-NMI	Avg-NMI
CSPA-METIS	0.740	**4.3**	9.4	6.1	7.9	0.699	0.601	0.648
CSPA-SPEC	0.743	4.4	9.4	6.1	7.9	0.699	0.601	0.648
CBPA-METIS	0.723	4.8	9.4	6.1	7.9	0.699	0.601	0.648
CBPA-SPEC	0.743	4.4	9.4	6.1	7.9	0.699	0.601	0.648

Table 5.7: Results on Vote data

	Ens-NMI	Ens-ER	Max-ER	Min-ER	Avg-ER	Max-NMI	Min-NMI	Avg-NMI
CSPA-METIS	0.473	14.0	17.7	6.9	13.7	0.640	0.345	0.447
CSPA-SPEC	0.449	**13.5**	17.7	6.9	13.7	0.640	0.345	0.447
CBPA-METIS	0.473	14.0	17.7	6.9	13.7	0.640	0.345	0.447
CBPA-SPEC	0.439	14.2	17.7	6.9	13.7	0.640	0.345	0.447

Chapter 6: Semi-Supervised Clustering

6.1 Introduction

In Chapters 3 and 4, we described the effectiveness and benefits of our consensus functions, which overcome the problems inherent to individual clustering algorithms. Each of our consensus functions is based on the assumption that no prior knowledge is available, as this is the condition for many research problems, and therefore an important matter to be addressed.

However, in some cases, limited information is indeed available to the end user. The possibility of leveraging this knowledge has attracted the attention of many researchers, who attempt to incorporate the available information as constraints into the clustering process with the aim of improving the results. The problem these researchers face is how to leverage this information. Although the presence of prior knowledge would seem to dictate its use, the means of achieving this goal are unclear.

We explore three different, but related approaches:

1. embed constraints into the consensus clustering to improve the quality of the ensemble.

2. embed constraints within individual clustering algorithms

3. embed constraints within individual clustering algorithms, and bootstrapping of constraints driven by ensembles.

We will investigate each approach in details in this Chapter. Our achievements in this area are:

- Improved effectiveness of our clustering ensembles achieved by enforcing constraints during the partitioning process when knowledge is available from the end-user.

- Improved effectiveness of the LAC algorithm achieved by using constraints to both refine the initialization process and the partitioning dynamic.

- Bootstrapping of constraints from multiple data partitions, without the intervention of an oracle.

6.2 Constraint Identification

6.2.1 Selecting Informative Constraints

A number of semi-supervised clustering algorithms have been proposed [66,67]. However, most of these techniques construct must-link and cannot-link constraints by first randomly selecting pairs of points, and then querying the oracle expert for information about their relationship. Although this method is relied on by many researchers, it has the liability of not improving the clustering process to its fullest potential. Because the selection process is random, and does not seize on associations available in the raw data, this method neglects a very important source of information. It tends to reinforce existing relationships, rather than provide a strong rationale for labeling difficult data.

To avoid these limitations, we follow the approach described in [29] to generate constraints. Because they are formed using associations available in the raw data, these constraints are stronger and more informative. The authors base their method on the relationships between intra-cluster and inter-cluster data points. The information provided from these relationships becomes must-link and cannot-link constraints.

The authors begin by mapping the results of k-means to a co-association matrix. The algorithm is then run ν times, and the resulting ν co-association matrices are averaged into a final matrix T. With a sufficient number of base clusterings, information about a pairwise relationship between two data points becomes apparent in the final matrix T.

The entry T_{ij} indicates the portion of the ν clusterings in which two data points $\mathbf{x}_i$ and $\mathbf{x}_j$, were assigned to the same cluster. A value of $T_{ij} = 1$ indicates that the points were assigned to the same cluster in each of the ν matrices, and therefore represents a very high probability that the points belong to the same class. A value of $T_{ij} = 0$, on the other hand, indicates that the points were not assigned to the same cluster in any of the ν matrices, and therefore represents a very low probability that the points belong to the same cluster and should be placed in different cluster.

However, if $T_{ij} \approx 0.5$, it is not clear whether the corresponding two points should belongs to the same cluster or not. Thus, querying on the underlying relationship between such data would be partiality informative. Two threshold values t_m and t_c are chosen to identify useful constraints. In particular pairs $(\mathbf{x_i}, \mathbf{x_j})$ such that $t_c < T_{ij} < t_m$.

Since this method proved to be effective, we adapted it for the selection of our constraint sets. We run the LAC algorithm m times. Each partition ν gives a co-association matrix T_ν, from which we compute the average co-association matrix $T = \frac{1}{m}\sum_{\nu=1}^{m} T_\nu$. We then select all pairs of points $(\mathbf{x}_i, \mathbf{x}_j)$, such that $T_{ij} \in [t_c, t_m]$. This selection mechanism allows us to identify pairs of points for which there exist great uncertainty on their clustering. Thus, querying an oracle about the underlying relationships adds valuable information which is not available from the data alone.

This process generates two constraint sets (M, C), where M corresponds to the set of must-link constraints, and C corresponds to the set of cannot-link constraints.

We use these constraints to form a chunklet graph as discussed in the next section. The pseudo code for the imputation of constraints is given in Algorithm 6.

Algorithm 6 Identify Imputed Constraint sets (M,C) (IM-Constraint) Algorithm

Input: m partitions of n data points

1. For each partition $\nu = 1, \ldots, m$:

   - Build the Co-Association matrix T_ν of size $n \times n$

$$T_{\nu_{ij}} = \begin{cases} 1 & \text{if } \mathbf{x}_i, \mathbf{x}_j \text{ are in the same cluster in partition } \nu \\ 0 & \text{if } \mathbf{x}_i, \mathbf{x}_j \text{ are in different clusters in partition } \nu \end{cases}$$

2. Form the final Co-Association Matrix T from all T_ν where $\nu = 1, \ldots, m$:

$$T = \frac{1}{m} \sum_{\nu=1}^{m} T_\nu$$

3. Select all pairs (or a random sample)$(\mathbf{x}_i, \mathbf{x}_j)$ s.t $T_{ij} \in [t_c, t_m]$
4. Query the oracle for the selected pairs $(\mathbf{x}_i, \mathbf{x}_j)$
5. Construct constraint sets (M, C)

Output: The resulting constraint sets (M, C)

6.2.2 Chunklet Graph

In devising our semi-supervised clustering methods, we have decided to build on chunklets. Chunklets are an efficient mean of grouping points, and provide solid information.

A chunklet is a group of points that belong to the same cluster, although the identity of the cluster is unknown [10]. The size of the chunklet is equal to the number of points it contains, so for chunklet $\triangle = (\mathbf{x}_1, \mathbf{x}_2)$, the size is $| \triangle |$ =2.

Each chunklet is formed by must-link constraints (M) obtained from the oracle expert. These pairwise constraints are a realistic path for building groups, as labeled

data is likely inaccessible. So, if the oracle imposes a must-link between points $\mathbf{x}_1$ and $\mathbf{x}_2$, then chunklet $\triangle_1 = (\mathbf{x}_1, \mathbf{x}_2)$ is formed.

Following the formation of chunklets through must-link constraints, a transitive closure process, in which chunklets are merged, is initiated. For example, if there is a must-link constraint between $(\mathbf{x}_1, \mathbf{x}_2)$ and $(\mathbf{x}_1, \mathbf{x}_3)$ then by using transitive closure the chunklet $\triangle_2 = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$ is formed.

Once chunklets are formed and all transitive closures completed, a graph is created using the cannot-link constraints imposed by the oracle expert. These constraints prevent the assignment of some chunklets to the same cluster. If, for example, there is a cannot-link constraint between the pair $(\mathbf{x}_3, \mathbf{x}_5)$ and we have the chunklet $\triangle_3 = (\mathbf{x}_4, \mathbf{x}_5)$, then our previously cited chunklet $\triangle_2 = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$ will be prevented from the assignment to the same cluster as chunklet $\triangle_3$.

The cannot-link constraint is represented on the graph as an edge between two vertices where each vertex corresponds to one chunklet. In this way the entire chunklet graph is obtained, where each chunklet is a vertex, and each cannot-link constraint is an edge. Edges indicate that the corresponding vertices (chunklets) should be assigned to different clusters.

The chunklet graph $G_{ch} = (V, E)$ is constructed, where V is a set of vertices (or chunklets) constructed from the must-link constraints M, and $|V|$ is the total number of chunklets. E is the set of edges, and an edge E_{ij} exists between vertices (chunklets) v_i and v_j iff there exist $\mathbf{x}_i \in v_i$, $\mathbf{x}_j \in v_j$ such that $(\mathbf{x}_i, \mathbf{x}_j) \in C$. The pseudo code of our chunklet graph construction is presented in Algorithm 7.

6.2.3 Chunklet Initialization

As with any clustering algorithm, such as the popular method k-means, the use of good initial centroids will have an effect on the final partition. As in [13], the

Algorithm 7 Chunklet Graph Algorithm

Input: constraint sets (M, C)

1. Compute the transitive closure for all the must-link constraints M
2. Build chunklets v_i using the must-link constraints
3. Construct chunklet graph $G_{ch} = (V, E)$: where V corresponds to the set of chunklets and $E_{ij} = 1$ iff there exist $\mathbf{x}_i \in v_i$, $\mathbf{x}_j \in v_j$ s.t $(\mathbf{x}_i, \mathbf{x}_j) \in C$

Output: The resulting Graph G_{ch}

use of prior knowledge in the form of labeled data is helpful in constructing a good initialization. Because labeled data is rarely accessible, we make use of the constraint-based chunklet graph to identify initial centroids.

Once we have constructed the graph, we build the initial centroids using the vertices, with cannot-link constraints between them. Thus, we select the vertices v_i and v_j such that $E_{ij} = 1$. We then obtain the mean vectors of the points contained in each corresponding chunklet. These mean vectors become the initial centroids.

If the number of selected chunklets is less than k (the number of desired clusters), we choose as additional initial centroids the points that are the farthest from the already chosen centroids. The selection is iterated until we reach k initial centroids.

6.2.4 Chunklet Assignment

Chunklet assignment is the process of assigning vertices (chunklets) in the graph to the appropriate centroid without violating any of the must-link or cannot-link constraints. We consider each chunklet to be a group of points and assign all points in the chunklet to the closest centroid.

Let us assume we use LAC as clustering algorithm. All points in each chunklet are assigned to the centroid that minimizes the sum of the weighted squared distance between them and the centroid. Given a vertex (chunklet) v_i, we calculate

the weighted Euclidean distances between all the points $\mathbf{x}_j \in v_i$ and each centroid $\mathbf{c}_l$ where $l = 1, \ldots, k$, and look for the centroid $\mathbf{c}_k$ that satisfies $\mathbf{c}_k = argmin_l(d(v_i, \mathbf{c}_l))$, where $d(v_i, \mathbf{c}_l) = \sum_{j=1}^{|v_i|} \sqrt{\sum_{s=1}^{D} w_{ls}(x_{js} - c_{ls})^2}$. D is the dimensionality of the data, and $\mathbf{w}_l$ is the weight vector associated with centroid $\mathbf{c}_l$.

We need to satisfy all given constraints in the assignment process by assigning vertices in the graph that have at least one or more edges to an appropriate centroids, and keep track of such assignments. In the following, we discuss three situations that require different centroid assignment strategies.

Case 1 An isolated chunklet v_i that does not have an edge in the graph G_{ch}. We assign this chunklet to the closest centroid $\mathbf{c}_l$ that minimizes the sum of the weighted squared distances between all the chunklet's points ($\mathbf{x}_j \in v_i$) and the centroid itself.

Case 2 A chunklet v_i that has at least one neighbor in the graph G_{ch}, and none of its neighbors have been assigned to any centroid. In this case, as before, we assign v_i to the closest centroid.

Case 3 A chunklet v_i that has at least one neighbor in the graph G_{ch} that has been assigned to a centroid. We construct the set of centroids S_c, to which the neighboring nodes of v_i have been assigned. We then consider each centroid $\mathbf{c}_u$, where $\mathbf{c}_u \notin S_c$, and find the closest centroid to v_i among them.

The process of treating each chunklet as a bulk of points, and then taking the weighted Euclidean distance to identify the appropriate centroids will aid in making reliable assignments. The pseudo code of the chunklet assignment procedure is illustrated in Algorithm 8.

Algorithm 8 Chunklet Assignment Algorithm

Input: Chunklet Graph G_{ch}, centroids $\{\mathbf{c}_1, \ldots, \mathbf{c}_k\}$, and weights $\{\mathbf{w}_1, \ldots, \mathbf{w}_k\}$

1. Create a matrix $O = zeros(u, k)$, where u is the total number of points in all chunklets of G_{ch}.

2. For each vertex $v_i \in G_{ch}$, we consider three cases:

   (a) Chunklet v_i does not have a neighbor in the graph G_{ch}
   Assign chunklet v_i to the closest centroid:

      i. $\mathbf{c}_k = argmin_l(d(v_i, \mathbf{c}_l))$, s.t.
      $d(v_i, \mathbf{c}_l) = \sum_{x_j \in v_i} \sqrt{\sum_{s=1}^{D} w_{ls}(x_{js} - c_{ls})^2}$

      ii. $\forall\ (\mathbf{x}_j \in v_i)$, set $O_{j,k} = 1$

   (b) Chunklet v_i does not have a neighbor that has been assigned to a centroid.

      i. Assign chunklet v_i to the closest centroid $\mathbf{c}_k$ as above.

      ii. $\forall\ (\mathbf{x}_j \in v_i)$, set $O_{j,k} = 1$

   (c) Chunklet v_i has at least one neighbor that has been assigned to a centroid.

      i. Construct the set of centroids, S_c, to which the neighboring chunklets have been assigned.

      ii. Find the closest centroid, $\mathbf{c}_k$, to chunklet v_i as described above, satisfying the condition $\mathbf{c}_k \notin S_c$.

      iii. $\forall\ (\mathbf{x}_j \in v_i)$, set $O_{j,k} = 1$

Output: The resulting matrix O

6.3 Constrained Locally Adaptive Clustering (CLAC)

6.3.1 Introduction

A wide range of research have been proposed for semi-supervised clustering. These algorithms often improve the clustering performance with respect to unsupervised clustering methods. Most of these techniques are capable of finding partitions for low dimensional dataset. However, as the dimensionality of the data increases, discovering meaningful clustering solutions become a difficult task even with the help of limited supervision. In this section, we address the high dimensionality problem by incorporating the side information in the initialization and re-iterative process of the LAC (2.3.3) algorithm. LAC is an effective subspace clustering algorithm that associates to each cluster a weight vector, whose values capture the relevance of features within the corresponding cluster. LAC avoids the risk of loss of information due to global dimensionality reduction techniques by considering all features, but properly weight each feature. Incorporating side information into LAC algorithm makes it a subspace clustering that adhere to the user's preferences. We call this technique CLAC (Constrained Locally Adaptive Clustering).

6.3.2 The CLAC Algorithm

Constrained Locally Adaptive Clustering (CLAC) is a soft feature selection procedure that integrates constrained learning through the refining of initial centroids, and during successive iterations. CLAC assigns weights to features according to the local correlations of data. Dimensions along which data are loosely correlated receive a

small weight, which has the effect of elongating distance along that dimension. Features along which data are strongly correlated receive a large weight, which has the effect of constricting distances along that dimension. CLAC produces weighted clusters without violating any must-link or cannot-link constraints (under the assumption that the satisfaction of all constraints is feasible).

In order to generate informative constraints to be incorporated into CLAC, we used the same approach as described in Section 6.2.1. Once we have identified the proper constraints, we build the chunklet graph as illustrated in Section 6.2.2. We incorporate the graph into CLAC as follows. CLAC selects the initial centroids using the chunklet graph as described in Section 6.2.3. Overall, this initialization procedure is able to take into account the side information provided, to obtain cluster representatives that can lead to a good initial partitioning of the data.

In addition, CLAC embeds the constraints during each iteration. CLAC incorporates the given constraints during each update of the centroids without violating any cannot-link constraints. This is achieved using the procedure described in Section 6.2.4. This assignment strategy ensures that at each iteration the data partition discovered satisfies the user's constraints. The process is iterated until convergence.

CLAC partitions the data into k clusters by assigning each data point to the closest centroid using a weighted Euclidean distance. However CLAC treats points in the chunklets as a group by assigning each chunklet to the closest centroid that minimizes the sum of the squared weighted distances between all the points in each chunklet and each centroid. It also ensures assigning chunklets connected by an edge to different centroids.

CLAC selects the initial centroids from the chunklet graph. As LAC, CLAC initializes all weights to $1/D$. The initial partition is obtained by using the chunklet assignment algorithm outlined in Algorithm 8. Points which are not contained in

any of the chunklets, are assigned to the closest centroid according to the weighted Euclidean distance. Weights are updated according the same equation (2.4)derived for LAC , the partition of the data is recomputed (again making use of constraints) and centroids are updated. The procedure is iterated until convergence. As LAC, CLAC requires the h parameter in input. The CLAC algorithm is summarized in Algorithm 9.

6.4 Weighted Clustering Ensembles with Limited Prior Knowledge

6.4.1 Introduction

Here we propose an extension of the applicability and effectiveness of our clustering ensemble discussed in Chapter (3), to situations where some knowledge is available from the end user. Our newly proposed method combines a clustering ensemble's ability to overcome the ill-posed nature of clustering with semi-supervised clustering's ability to leverage an end user's knowledge. Our technique enforces knowledge-based constraints during the partitioning of each component clustering to improve the quality of the ensemble.

Our approach is motivated by the work of [18] where they used ensemble to produce final partition then use labeled data to assign cluster to class. Their approach aim on the improvement of semi-supervised classification to label new coming point; whereas our approach aim to enhance the clustering ensembles technique by enforcing constraints during the partitioning process.

Enforcing constraint sets provided by the end user would improve the performance of the ensemble. This prior knowledge carries information on the underlying structure

Algorithm 9 CLAC Algorithm

Input: n points $\mathbf{x} \in \Re^D$, k, and h.

1. (M, C)= IM-Constraint (m partitions)
2. $S_1 = \emptyset, \ldots, S_k = \emptyset$
3. (G_{ch})= Chunklet-Graph (M, C);

Chunklet Initialization

$\forall\ v_i \in G_{ch}$ such that $E_{ij} = 1$ for some j

Compute the mean of points in v_i and assign it as initial centroid;

Set z = number of selected centroids;

while $(z < k)$

Select the farthest point from the already selected centroids, and assign it as initial centroid;

Let $\{\mathbf{c}_1, \ldots, \mathbf{c}_k\}$ be the resulting centroids;

4. Set $w_{sj} = 1/D$, for each centroid $\mathbf{c}_j$, $j = 1, ..., k$ and each feature $s = 1, ..., D$;
5. O= Chunklet-Assignment $(G_{ch}, \{\mathbf{c}_1, \ldots, \mathbf{c}_k\}, \{\mathbf{w}_1, \ldots, \mathbf{w}_k\})$

For each $v_i \in G_{ch}$, let t_i be the assigned cluster centroid (as determined by the chunklet assignment procedure)

$\forall \mathbf{x} \in v_i$, $S_{t_i} = S_{t_i} \bigcup \{\mathbf{x}\}$;

6. For each centroids $\mathbf{c}_j$, and for each point $\mathbf{x} \notin v_i, \forall_i$

$S_t = S_t \bigcup \{\mathbf{x} | t = argmin_l L_w(\mathbf{c}_l, \mathbf{x})\}$

where $L_w(\mathbf{c}_l, \mathbf{x}) = (\sum_{s=1}^{D} w_{ls}(c_{ls} - x_s)^2)^{1/2}$;

7. **Compute new weights**

For each centroid c_j, and for each feature s:

Set $X_{js} = \sum_{x \in S_j} (c_{js} - x_s)^2 / |S_j|$;

Set $w_{js} = \frac{exp(-X_{js}/h)}{\sum_{s=1}^{D} exp(-X_{js}/h)}$;

8. O= Chunklet-Assignment $(G_{ch}, \{\mathbf{c}_1, \ldots, \mathbf{c}_k\}, \{\mathbf{w}_1, \ldots, \mathbf{w}_k\})$

For each $v_i \in G_{ch}$, let t_i be the assigned cluster centroid (as determined by the chunklet assignment procedure)

$\forall \mathbf{x} \in v_i$, $S_{t_i} = S_{t_i} \bigcup \{\mathbf{x}\}$;

9. For each centroids $\mathbf{c}_j$, and for each point $(\mathbf{x} \notin v_i, \forall_i)$

$S_t = S_t \bigcup \{\mathbf{x} | t = argmin_l L_w(\mathbf{c}_l, \mathbf{x})\}$

where $L_w(\mathbf{c}_l, \mathbf{x}) = (\sum_{s=1}^{D} w_{ls}(c_{ls} - x_s)^2)^{1/2}$;

10. **Compute new centroids**

Set $c_j = \frac{\sum_x x 1_{S_j}(x)}{\sum_x 1_{S_j}(x)}$, for each $j = 1, ..., k$, where $1_S(.)$ is the indicator function of set S;

11. Iterate 5-10 until convergence (no change in cluster assignment).

Output: The partition $S = \{S_1, \ldots, S_k\}$

of the data on which the components of the ensemble were not able to agree. By enforcing such constraints at the components' level, we ensure that the corresponding structure they represent is carried out into the consensus function, and therefore into the consensus partition. Thus, we aim to produce a robust and stable solution of the given data that adheres to the user's preference.

6.4.2 Constrained-Weighted Bipartite Partitioning Algorithm (C-WBPA)

Our goal is to generate robust and stable solutions via a consensus clustering method that makes use of prior knowledge under the form of must-link (two points must be assigned to the same cluster) and cannot-link (two points must be assigned to different clusters) constraints. We generate contributing clusterings by running the LAC algorithm multiple times by changing the h parameter. The process of our approach is as follows.

We generate imputed constraints using the approach described in Section 6.2.1. Once we have identified the proper constraints, we build the chunklet graph as illustrated in Section 6.2.2. We then incorporate the graph into clustering ensembles components without violating any cannot-link constraints using the procedures described in Section 6.2.4. The assignment strategy is applied for each run of the LAC algorithm, changing the h parameter in each run.

Since LAC produces weighted clusters, we use this information to assign the vertices in the chunklet graph to the closest centroid. For each partition of LAC we assign each chunklet to the closest centroid which minimizes the sum of the squared weighted distances between all the points in each chunklet and each centroid. This assignment will produce a matrix O of values [0,1], where the number of rows is equal

to the number of points in the chunklet graph, and the number of columns is equal to the number of k clusters. The entry $(ij) = 1$ means that the point i is most likely belongs to cluster j. We also ensure the assignment of chunklets connected by an edge to different centroids. However, for points $\mathbf{x}_i$ not involved in the constraint sets, and for each clustering $\nu = 1, \ldots, m$ we follow our consensus clustering approach WBPA (introduced in Section 3.2.2).

We then initialize a matrix N to zeros values, where the rows are equal to the total number of n points, and the columns are equal to the number of k clusters. We then start filling the matrix N for each data point $\mathbf{x}$. For the points participating in the chunklet graph, we retrieve their row value from the i-th row matrix O and assign it to N (i-th row of N). For points not involved in the constraint set, we extract their probability vectors P_i and assign it to N_i. We follow the above procedure for each partition. Finally, we construct the following AN matrix:

$$AN = \begin{pmatrix} N^1 & N^2 & \ldots & N^m \end{pmatrix} \tag{6.1}$$

For the aggregation step, we rely on the consensus clustering approach WBPA (introduced in Section 3.2.2). The steps of the algorithm, which we call C-WBPA (Constrained, Weighted Bipartite Partitioning Algorithm), are summarized in Algorithm 10.

6.4.3 Experimental Design

In our experiments, we used eight real datasets. The characteristics of all datasets are given in Table 6.1. Iris, Breast, Letter(A,B), Wine and Ionosphere are from the UCI Machine Learning Repository [8]. WDBC is the Wisconsin Diagnostic Breast Cancer dataset [52]. Ling-Spam and 20Newsgroups are two high dimensional text

Algorithm 10 Constrained-Weighted Bipartite Partitioning Algorithm

Input: n points $\mathbf{x} \in R^D$, and k

1. Run LAC m times with different h values. Obtain the m partitions: $\{\mathbf{c}_1^\nu, \ldots, \mathbf{c}_k^\nu\}$,$\{\mathbf{w}_1^\nu, \ldots, \mathbf{w}_k^\nu\}$, $\nu = 1, \ldots, m$
2. (M, C)= IM-Constraint $(m - partitions)$
3. (G_{ch})= Chunklet-Graph $(M, C,)$
4. For each partition $\nu = 1, \ldots, m$:
   (a) O^ν= (chunklet-Assg) $(G_{ch}, \{\mathbf{c}_1^\nu, \ldots, \mathbf{c}_k^\nu\}, \{\mathbf{w}_1^\nu, \ldots, \mathbf{w}_k^\nu\})$
   (b) $\forall$ $\mathbf{x}_i$ not involved in any constraints
      i. Compute $d_{il}^\nu = \sqrt{\sum_{s=1}^{D} w_{ls}^\nu (x_{is} - c_{ls}^\nu)^2}$
      ii. Set $D_i^\nu = \max_l\{d_{il}^\nu\}$
      iii. Compute $P(C_l^\nu|\mathbf{x}_i) = \frac{D_i^\nu - d_{il}^\nu + 1}{kD_i^\nu + k - \sum_l d_{il}^\nu}$
      iv. Set $P_i^\nu = (P(C_1^\nu|\mathbf{x}_i), P(C_2^\nu|\mathbf{x}_i), \ldots, P(C_k^\nu|\mathbf{x}_i))^t$
   (c) Initialize $N^\nu = zeros$
      $\forall$ $\mathbf{x}_i$ not involved in constraints
         $N^\nu = P_i^\nu$, [i-th row of N^ν is set equal to P_i^ν]
      $\forall$ $\mathbf{x}_i$ in constraints
         $N^\nu = O^\nu$, [i-th row of N^ν is set equal to O^ν]
5. Construct the matrix AN as in (6.1)
6. Construct the bipartite graph $G = (V, E)$, where $V = V^C \cup V^I$, $|V^I| = n$ and $V_i^I \equiv \mathbf{x}_i$, $|V^C| = km$ and $V_j^C \equiv C_j$ (a cluster of the ensemble). Set $E(i, j) = 0$ if V_i and V_j are both clusters or both instances. Set $E(i, j) = AN(i - km, j) = E(j, i)$ if V_i and V_j represent an instance and a cluster
7. Run METIS on the resulting graph G

Output: The resulting k-way partition of the n vertices in V^I

Table 6.1: Characteristics of the datasets

Dataset	k	D	n (points-per-class)
Iris	3	4	150 (50-50-50)
WDBC	2	31	424 (212-212)
Breast	2	9	478 (239-239)
Letter(A,B)	2	16	1555 (789-766)
Wine	3	13	178 (59-71-48)
Ionosphere	2	33	239 (126-113)
20Newsgroups(ele-med)	2	321	1971 (981-990)
Ling-Spam	2	350	906 (453-453)

datasets. The text documents in each dataset were preprocessed by eliminating stop words (based on a stop words list) and stemming words to their root source. As feature values in the vector space model we have used the frequency of the terms in the corresponding document. To reduce the dimensionality of the data, we followed the procedure presented in [42].

Ling-Spam is a mixture of spam messages (453) and messages (561) sent via the linguist list, a moderated (hence, spam-free) list about the profession and science of linguistics. The original size of the dictionary is 24627. After processing the data as described above, the dictionary size was reduced to 350. 20 Newsgroups is a collection of 20,000 messages collected from 20 different netnews newsgroups. One thousand messages from each of the 20 newsgroups were chosen at random and partitioned by newsgroups name. In our experiments we consider the categories medical (990) and electronics (981). The original size of the dictionary is 24,546; after processing the data, the dictionary size was reduced to 321.

Since METIS [43] requires balanced datasets, we performed random sampling on Breast, WDBC, Ionosphere, and Ling-Spam. In each case, we sub-sampled the most populated class: from 357 to 212 for WDBC, from 444 to 239 for Breast , from 225 to 113 for Ionosphere, and from 561 to 453 for LingSpam. Also For the Letter dataset,

we used the classes "A" and "B" (balanced).

We compared our constrained bipartite partitioning algorithm (C-WBPA), with other semi-supervised clustering approaches: COP-Kmeans [66], and Seeded-COP-Kmeans. We ran COP-Kmeans ten times with random initialization. Seeded-COP-Kmeans was initialized using the initialization procedures described in Section 6.2.3. We also compared C-WBPA with the Constrained Locally Adaptive Clustering (CLAC) algorithm presented in Section 6.3. CLAC was ran multiple times for different value of the h parameter.

Evaluating the quality of clustering is in general a difficult task. Since class labels are available for the datasets used here, we evaluate the results by computing the error rate. The error rate is computed according to the confusion matrix.

6.4.4 Analysis of the Results

We input the same set of constraints used for C-WBPA into COP-Kmeans, Seeded-COP-Kmeans, and also into the CLAC to have a fair comparison. (As the value of k, we input for all the techniques the actual number of classes in the data.) Figure 6.1 plots the error rate (%) and standard deviations for an increase number of constraints for each dataset. Each figure clearly shows the improvement of our (C-WBPA) algorithm with respect to the other techniques. The trends of the error rate clearly depends on the data distribution.

We notice the smooth trends of our technique (C-WBPA) with an increasing number of pairwise constraints.

Our technique achieves the lowest error rate with small number of constraints in most cases. This is because the ensemble is able to filter superior structures in the data and if limited information is available, our C-WBPA will achieve a good result. This characteristic makes our approach valuable asset to be used with very limited

knowledge with no need to query oracle for more information.

CLAC also achieve smooth trends with the increase number of pairwise constraints, this because CLAC depends on the setting of h parameter. CLAC achieves improvement result with respect to the LAC algorithm, but not for all values of the h parameter. For example, for the Breast cancer dataset we notice the large value of standard deviations for each input of constraints. This indicates a great variability of the error rate between individual components from the mean. Further investigation leads to notice that CLAC achieves good result for various value of the h parameter, but not for the case where $h = 10$. For $h = 10$ CLAC achieves an error rate of 34.94% even with the increase number of pairwise constraints, which made the average of the error rate very bad with respect to other techniques.

Also, for the WDBC dataset we notice the increase rate of the error and standard deviations with the increase number of pairwise constraints. This is clear from the figure, especially when the input constraints reaches to 50 of the number of pairwise constraints or more. The increase of the error rate along with the increase of standard deviations make CLAC not stable for some value of the h parameter and indicate a great scatter of the individual components with a wide range. Therefore, an increase of the number of pairwise constraints is not necessary would reduce the error rate and the standards deviations for different value of the h parameter.

Increasing the number of constraints made available to each component may induce a high degree of correlation between them, causing diversity to decrease. We proved in Section 4.3.1 that a high diversity in clustering results contributes to a high accuracy in the aggregation results. This phenomenon might be the reason for the stable error rate of C-WBPA for increasing constraints. We emphasize the large improvements obtained by both C-WBPA and CLAC for high dimensional datasets

(20Newsgroups, Ling-Spam) we tested. This is because the LAC and CLAC algorithm techniques are designed to handle data with high dimensionality, while any variation of k-means tends to break down for datasets with high dimensionality.

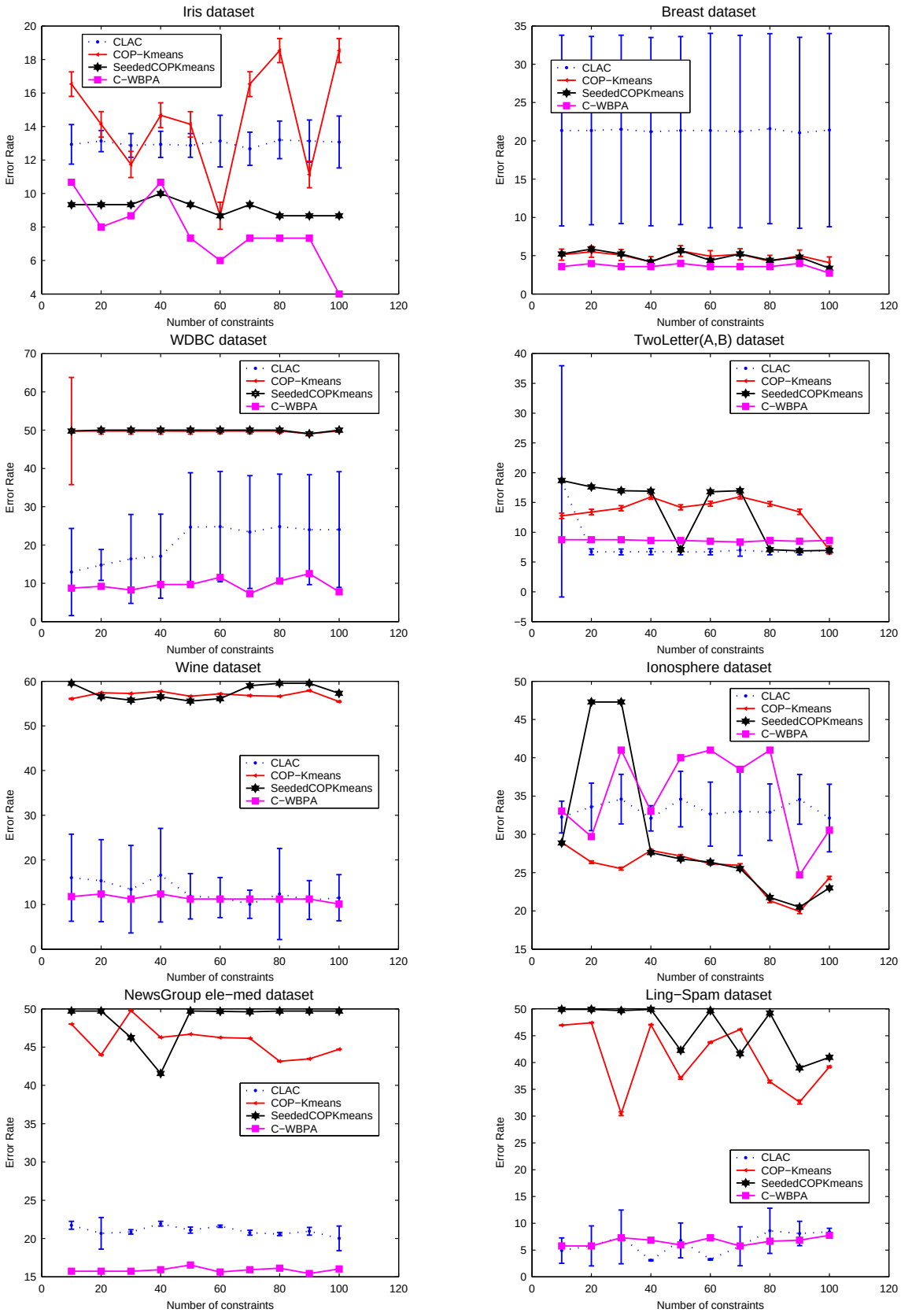

Figure 6.1: Constrained Clustering Ensemble Results

6.5 Bootstrapping of Constraints: Penta-training

6.5.1 Introduction

In our penta-training framework we combine clustering ensembles with semi-supervised clustering. We use the ensemble framework to bootstrap informative constraints directly from the data, and from the different clustering components. Our approach is well suited for problems where the information available from an external source (e.g., domain expert) is very limited. We also demonstrate the feasibility of our technique to situations where prior knowledge is absent. Our work is motivated by co-training [14] and tri-training [70]. As co-training and tri-training, we leverage the ensemble methodology to perform semi-supervised learning. While co-training and tri-training use classifiers as learning components, and propagate labels among them, our technique uses a collection of clusterings (five, from which the name *Penta-Training*) to derive constraints. To the best of our knowledge, this is the first attempt of its kind. We use CLAC as the basic component of our penta-training framework, where the individual clusterings are iteratively refined using constraints generated during the penta-training process.

6.5.2 Penta-Training Algorithm

Penta-Training assembles multiple (five) clusterings obtained by CLAC, and bootstraps constraints to improve the quality of the components, and ultimately of the ensemble.

Given an initial collection of constraints (M, C), we run CLAC five times with different values of the h parameter. Each run of CLAC is provided with the entire data set and the entire constraint set. We obtain five clusterings of the data. Penta-training leverages the consensus achieved across such partitions to bootstrap

and propagate constraints: we look for pairs of points on which four (out of the five) clusterings agree (and the fifth disagree), i.e., all four clusterings group the two points together, or separately. In the first case, a must-link constraint is generated for the fifth component; in the second case, a cannot-link constraint is generated. Once all constraints for a given component have been generated, they are added to the current set of constraints of that component, and CLAC is re-run. The process is iterated for all combinations of four components, until no change in all five clusterings is observed. To ensure that only relevant constraints are propagated, we rank the candidate constraints, and use only the top ranked ones. In particular, for each candidate must-link constraint $(\mathbf{x}_n, \mathbf{x}_m)$, we compute the four weighted Euclidean distances, using the corresponding weights of the clusters the two points $\mathbf{x}_n$ and $\mathbf{x}_m$ are assigned to, and compute their average. The average distances are then sorted in ascending order. We select the top ranked pairs (with smallest distances) as must-link constraints for the fifth component. We proceed similarly for the cannot-link contraints. For each candidate cannot-link constraint we compute their Euclidean distance (note that in this case, four clusterings place the points in different clusters, and therefore there is no single weight vector associated with them). We then sort the distances in descending order. We select the top ranked pairs (with largest distances) as cannot-link constraints for the fifth component. In our experiment, at each iteration of penta-training, we select the top five ranked must-link and the top five ranked cannot-link constraints.

We observe that penta-training can be applied also when no constraints are initially available. In this case, we start building the ensemble by simply running the original LAC algorithm (with no side-information) using different values of h. As constraints are bootstrapped during the rounds of penta-training, LAC is substituted by CLAC. We test this scenario as well in our experiments.

Table 6.2: Characteristics of the datasets

Dataset	k	D	n (points-per-class)
Iris	3	4	150 (50-50-50)
WDBC	2	31	424 (212-212)
Breast	2	9	478 (239-239)
Wine	3	13	144 (48-48-48)
Ionosphere	2	33	239 (126-113)

At convergence, we have available five partitions (precisely, each partition corresponds to k centroids, and corresponding weight vectors). We map the problem of finding a consensus function to a graph partitioning problem, by applying the WBPA (Weighted Bipartite Partitioning) algorithm presented in Section 3.2.2, which has been demonstrated to be effective. The WBPA algorithm takes into account not only how often points are grouped together across the clusterings, but also the degree of confidence of the groupings (by means of the weights). The Penta-Training algorithm is summarized in Algorithm 11.

6.5.3 Experimental Design

In our experiments, we used five real datasets. The characteristics of all datasets are given in Table 6.2. Iris, Breast, Wine and Ionosphere are from the UCI Machine Learning Repository [8]. WDBC is the Wisconsin Diagnostic Breast Cancer dataset [52].

The clustering ensemble algorithm WBPA uses METIS [43] to compute the k-way partitioning of a graph. Since METIS [43] requires balanced datasets, we performed random sampling on Breast, WDBC, Wine, and Ionosphere. In each case, we subsampled the most populated class: from 357 to 212 for WDBC, from 444 to 239 for Breast, from 59 to 48 and 71 to 48 for Wine, and from 225 to 113 for Ionosphere.

Algorithm 11 Penta-training Algorithm

Input: n points $\mathbf{x} \in \Re^D$, and k.
Run LAC algorithm m times for $\nu = 1, \ldots, m$
(M, C)= IM-Constraint $(m - partitions)$
(G_{ch})= Chunklet-Graph (M, C)
$S^{(h_i)}$= CLAC($\{\mathbf{x}\}$, k, h_i, M, C), for $i = 1, \ldots, 5$;
Let $T_n^{(h_i)} \in S^{(h_i)}$, be the set in partition $S^{(h_i)}$ to which $\mathbf{x}_n$ is assigned;
[*Initialization of constraints for each component*]
for $i = 1, \ldots, 5$
  $M^{(h_i)} = M, C^{(h_i)} = C$;
repeat
  for $l = 1, \ldots, 5$
  [*Bootstrapping of must-link constraints*]
    for every pair $(\mathbf{x}_n, \mathbf{x}_m) \notin M$
      if ($(T_n^{(h_l)} \neq T_m^{(h_l)})$ and $(\forall i \neq l,\, T_n^{(h_i)} = T_m^{(h_i)})$)
        Calculate the average weighted distance:
        $d(\mathbf{x}_n, \mathbf{x}_m) = \frac{1}{4} \sum_{i \neq l} (\sum_{s=1}^{D} w_{ts}(x_{n_s} - x_{m_s})^2)^{1/2}$;
        [$\mathbf{w_t}$ is the weight vector of the cluster the points $\mathbf{x}_n$ and $\mathbf{x}_m$ are assigned to]
      end if
    end for
    Sort above distances in ascending order;
    Select a percentage of top ranked pairs (with smallest distances), and add them to M_{h_l};
  [*Bootstrapping of cannot-link constraints*]
    for every pair $(\mathbf{x}_n, \mathbf{x}_m) \notin C$
      if ($(T_n^{(h_l)} = T_m^{(h_l)})$ and $(\forall i \neq l,\, T_n^{(h_i)} \neq T_m^{(h_i)})$)
        Calculate the Euclidean distance:
        $d(\mathbf{x}_n, \mathbf{x}_m) = \sum_{s=1}^{D} w_{ts}(x_{n_s} - x_{m_s})^2)^{1/2}$;
      end if
    end for
    Sort above distances in descending order;
    Select a percentage of top ranked pairs (with largest distances), and add them to C_{h_l};
    Run CLAC($\{\mathbf{x}\}$, k, h_l, M_{h_l}, C_{h_l});
  end for
until convergence [all five clusterings do not change]
Input the obtained five partitions (and corresponding weights) to WBPA presented in Section 3.2.2;
Output: Partition of the n data points into k clusters.

We tested our penta-training framework with two scenarios: in one case, a limited number of constraints is initially available; in the second case, no constraints are available. For the first scenario, to generate the initial set of constraints, we follow the procedure introduced in Section 6.2.1. For each dataset, the number of constraints generated is equal to 20% the number of data available. The obtained constraints are given in input to all five CLAC components. Each run of CLAC uses a different values of the h parameter. In our experiments we use the values $\{1, 3, 5, 7, 10\}$. According to our experience, this range of values provides in general diverse and accurate components. In the second case, when no initial constraints are available, we build the ensemble by running the LAC algorithm with the five values of h. As constraints are bootstrapped during the iterations of penta-training, LAC is substituted by CLAC.

At each iteration of penta-training, for each component, we bootstrap the top five ranked must-link constraints, and the top five ranked cannot-link constraints. We compare the following technniques:

- **LAC** [19]. We run the LAC algorithm five times for $h \in \{1, 3, 5, 7, 10\}$, and report average error rates and standard deviations.

- **CLAC**. We run the CLAC algorithm five times for $h \in \{1, 3, 5, 7, 10\}$, and report average error rates and standard deviations. We provide CLAC the set of constraints generated according to the procedure described in Section 6.2.1

- **Penta-Training** *with initial constraints.* We start penta-training with the same initial set of constraints we feed all competitive semi-supervised techniques.

- **Penta-Training** *without initial constraints.* In this case, no external constraints are used. The ensemble starts off in an unsupervised mode (LAC components), and constraints are bootstrapped in successive iterations from the data.

6.5.4 Analysis of the Results

Table 6.3: Penta-training accuracy results

Methods	Iris	Breast	WDBC	Ionosphere	Wine
LAC	14.13 ± 2.18	15.9 ± 11.78	19.76 ± 15.75	34.06 ± 4.20	15.16 ± 11.15
CLAC (20%)	13.06 ± 1.74	20.08 ± 13.15	16.69 ± 8.92	32.38 ± 3.08	15.69 ± 11.59
Penta-training (20%)	**10.67**	3.56	9.19	**31.38**	11.11
Penta-training (w/o const)	14	**3.14**	**8.73**	**31.38**	**9.03**

Error rates and standard deviations are reported in Table 6.3. In all five problems, penta-training provided the lowest error rate, or an error rate very close to the minimum. In some cases, penta-training provides huge improvements with respect to LAC, and CLAC. This indicates that the collaborative approach adopted by penta-training allows the bootstrapping of accurate and relevant constraints for the clustering process. In particular, as expected, the largest improvements are achieved when LAC and CLAC have large standard deviations (i.e., on Breast, WDBC, and Wine). In these cases, the components are diverse, and the ensembles become most effective. Also, In one dataset, Ionosphere, penta-training without constraints and penta-training with constraints achieved equally accurate results. Quite interesting is the fact that penta-training without initial constraints in most cases performs better than penta-training with initial constraints. This shows the efficacy of our data-driven and ensemble-driven constraints.

Chapter 7: Conclusion and Future Research

This chapter summarizes the dissertation, and suggests directions for future research.

7.1 Conclusions

We have discussed the challenges related to clustering and semi-supervised clustering, and presented novel techniques to address them.

In applying clustering methods, researchers have found problems arising from high dimensionality and parameter tuning. Our three algorithms make use of the ensemble methodology, which capture the common structures discovered by multiple clustering results, while averaging out emergent spurious structures. We apply our techniques to several real datasets, including high-dimensional text data, and categorical data.

The experimental results show that our clustering ensembles can provide solutions that are as good as or better than the best individual clustering. Furthermore, our results show that a high level of diversity is correlated with a high level of accuracy. Thus, provided that input clusterings are diverse, our weighted ensemble methods can provide robust and stable solutions.

We also developed three methodologies for semi-supervised clustering techniques. Our techniques embed side-information (in terms of pair-wise constraints) into clustering ensembles. The first technique enforces constraints during the partitioning process of each component to improve the quality of the overall ensemble, and achieve a result that adheres to the user preference. The second technique embeds constraints into the initialization and iterative phases of subspace clustering, thereby providing

a subspace semi-supervised clustering that can handle high dimensional data. The third technique, bootstraps constraints for the data without the intervention of an oracle expert.

The experimental results show that our semi-supervised clustering ensembles can provide solutions that are as good as or better than other semi-supervised clustering approaches.

7.2 Future Research

Because of its scope, there are many possibilities for extending the research described in this dissertation. Here we highlight directions for future work.

We have investigated the role of diversity in selecting ensemble components that can provide a high quality consensus clustering. Future work includes the definition of a selection mechanism that depends on the dataset and the ensemble components.

In addition, we have developed new and effective techniques in the area of semi-supervised clustering that make use of side-information in different ways. We foresee many possibilities to extend our work.

In our future work we will consider the design of a semi-supervised clustering method that embeds constraints in the final consensus function. Also, we will investigate techniques to embed constraints across different ensemble components.

For penta-training we would like to achieve an optimal trade-off between bootstrapping reliable constraints and maintaining a certain level of diversity among the components. One possibility, is to generate a large number of clustering components, and randomly select three partitions to bootstrap constraints.

Finally, the proposed techniques assume that the number of clusters to be found is given. A future research may venture into estimate the number of clusters using

the side-information provided by the end user.

Bibliography

Bibliography

[1] C. Aggarwal, J. Wolf, P. Yu, C. Procopiuc, and J. Park. Fast algorithms for projected clustering. In *Proceedings of ACM SIGMOD international conference on Management of data*, 1999.

[2] C. Aggarwal and P. Yu. Finding generalized projected clusters in high dimensional spaces. In *Proceedings of ACM SIGMOD international conference on Management of data*, 2000.

[3] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of ACM SIGMOD international conference on Management of data*, 1998.

[4] M. Al-Razgan and C. Domeniconi. Weighted clustering ensembles. In *Proceedings of SIAM International Conference on Data Mining*, 2006.

[5] M. Aldenderfer and R. Blashfield. Cluster Analysis. Sage Publications, 1984.

[6] K. Ali and M. Pazzani. Error reduction through learning multiple descriptions. *Machine Learning*, 24(3), 1996.

[7] P. Andritsos. *Scalable Clustering of Categorical Data And Applications*. PhD thesis, University of Toronto, Dep. of Computer Science, 2004.

[8] A. Asuncion and D.J. Newman. UCI machine learning repository, 2007.

[9] H. Ayad and M. Kamel. Finding natural clusters using multi-clusterer combiner based on shared nearest neighbors. In *Proceedings of Multiple Classifier Systems*, 2003.

[10] A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall. Learning distance functions using equivalence relations. In *Proceedings of 20th International Conference on Machine Learning*, 2003.

[11] D. Barbará and P. Chen. Using the fractal dimension to cluster datasets,. In *Proceedings of the the ACM-SIGKDD International Conference on Knowledge and Data Mining*, 2000.

[12] D. Barbará, Y. Li, and J. Couto. Coolcat: an entropy-based algorithm for categorical clustering. In *Proceedings of the eleventh international conference on Information and knowledge management*, 2002.

[13] S. Basu, A. Banerjee, and R. Mooney. Semi-supervised clustering by seeding. In *Proceedings of the Nineteenth International Conference on Machine Learning*, 2002.

[14] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the Workshop on Computational Learning Theory, Morgan Kaufmann Publishers*, 1998.

[15] L. Breiman. Bagging predictors. *Machine Learning*, 24(2), 1996.

[16] C. Cheng, A. Fu, and Y. Zhang. Entropy-based subspace clustering for mining numerical data. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, 1999.

[17] I. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2001.

[18] E. Dimitriadou, A. Weingessel, and K. Hornik. A mixed ensemble approach for the semi-supervised problem. In *Proceedings of the International Conference on Artificial Neural Networks*, 2002.

[19] C. Domeniconi, D. Gunopulos, S. Ma, B. Yan, M. Al-Razgan, and D. Papadopoulos. Locally adaptive metrics for clustering high dimensional data. *Data Mining and Knowledge Discovery Journal*, 14(1):63–97, 2007.

[20] C. Domeniconi, D. Papadopoulos, D. Gunopulos, and S. Ma. Subspace clustering of high dimensional data. In *Proceedings of SIAM International Conference on Data Mining*.

[21] C. Domeniconi and B. Yan. On error correlation and accuracy of nearest neighbor ensemble classifiers. In *Proceedings of SIAM International Conference on Data Mining*, 2005.

[22] S. Dudoit and J. Fridlyand. Bagging to improve the accuracy of clustering procedure. *Bioinformatics*, 19(9), 2003.

[23] X. Fern and C. Brodley. Random projection for high dimensional data clustering: A cluster ensemble approach. In *Proceedings of International Conference on Machine Learning*, 2003.

[24] X. Fern and C. Brodley. Solving cluster ensemble problems by bipartite graph partitioning. In *Proceedings of International Conference on Machine Learning*, 2004.

[25] A. Fred and A. Jain. Data clustering using evidence accumulation. In *Proceedings of International Conference on Pattern Recognition*, 2002.

[26] Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *Proceedings of International Conference in Machine Learning*, 1996.

[27] G. Gan and J. Wu. Subspace clustering for high dimensional categorical data. *SIGKDD Explor. Newsl.*, 6(2):87–94, 2004.

[28] D. Gondek and T. Hofmann. Non-redundant clustering with conditional ensembles. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2005.

[29] D. Greene and P. Cunningham. An ensemble approach to identifying informative constraints for semi-supervised clustering. In *Proceedings of the 18th European Conference on Machine Learning*, 2007.

[30] D. Greene, A. Tsymbal, N. Bolshakova, and P. Cunningham. Ensemble clustering in medical diagnostics. In *Proceedings of the 17th IEEE Symposium on Computer-Based Medical Systems*, 2004.

[31] S. Guha, R. Rastogi, and K. Shim. Rock: A robust clustering algorithm for categorical attributes. In *Proceedings of the 15th International Conference on Data Engineering*, 1999.

[32] S. Hadjitodorov, L. Kuncheva, and L. Todorova. Moderate diversity for better cluster ensembles. *Inofrmation Fusion*, 2005.

[33] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 1st edition edition, 2000.

[34] L. Hansen and P. Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10), 1990.

[35] Z. He, X. Xu, and S. Deng. Clustering mixed numeric and categorical data: A cluster ensemble approach. *ArXiv Computer Science e-prints*, 2005.

[36] Z. He, X. Xu, and S. Deng. Tcsom: Clustering transactions using self-organizing map. *Neural Processing Letters*, 22(3):249–262, 2005.

[37] T. Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.

[38] X. Hu. Integration of cluster ensemble and text summarization for gene expression analysis. In *Proceedings of IEEE Symposium on Bioinformatics and Bioengineering*, 2004.

[39] Z. Huang. Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data Mining and Knowledge Discovery*, 2(3), 1998.

[40] A. Jain, R. Duin, and J. Mao. Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1), 2000.

[41] A. Jain, M. Murty, and P. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3), 1999.

[42] N. Kang, C. Domeniconi, and D. Barbará. Categorization and keyword identification of unlabeled documents. In *Proceedings of the Fifth IEEE International Conference on Data Mining*, 2005.

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

[44] E. Kleinberg. Stochastic discrimination. *Annals of Mathematics and Artificial Intelligence*, 1, 1990.

[45] E. Kleinberg and T. Ho. Pattern recognition by stochastic modeling. In *Proceedings of the 3rd Workshop on Frontiers in Handwriting Recognition*, 1993.

[46] S. Kotsiantis and P. Pintelas. Recent advances in clustering: A brief survey. Technical report, Department of Mathematics University of Patras, 2004.

[47] S. Kullback and R. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22:79–86, 1951.

[48] L. Kuncheva and S. Hadjitodorov. Using diversity in cluster ensemble. In *IEEE International Conference on Systems, Man and Cybernetics*, 2004.

[49] L. Kuncheva and S. Hadjitodorov. Using diversity in cluster ensembles. In *Proceedings of International Conference on Systems, Man and Cybernetics*, 2004.

[50] L. Kuncheva, S. Hadjitodorov, and L. Todorova. Experimental comparison of cluster ensemble methods. In *Proceedings of International Conference on Information Fusion*, 2006.

[51] L. Kuncheva, S. Hadjitodorov, and L. Todorova. Experimental comparison of cluster ensemble methods. In *Proceedings of International Conference on Information Fusion*, 2006.

[52] O. Mangasarian and W. Wolberg. Cancer diagnosis via linear programming. *SIAM News*, 23(5):1–18, 1990.

[53] Q. Mei, D. Xin, H. Cheng, J. Han, and C. Zhai. Generating semantic annotations for frequent patterns with context analysis. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006.

[54] B. Minaei-Bidgoli, A. Topchy, and W. Punch. A comparison of resampling methods for clustering ensembles. In *Proceedings of the International Conference on Machine Learning; Models, Technologies and Applications*, 2004.

[55] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: analysis and an algorithm. In *Proceedings of Advances in Neural Information Processing Systems*, 2002.

[56] L. Parsons, E. Haque, and H. Liu. Subspace clustering for high dimensional data: a review. *SIGKDD Explorations, Newsletter of the ACM Special Interest Group on Knowledge Discovery and Data Mining*, 6(1):90–105, 2004.

[57] E. Pekalska. Dissimilariy representations in pattern recognition. In *PhD Thesis, Delft University of Technology, The Netherlands*, 2005.

[58] C. Rosenberg, M. Hebert, and H. Schneiderman. Semi-supervised self-training of object detection models. In *Proceedings of the Seventh IEEE Workshops on Application of Computer Vision*, 2005.

[59] M. Schroeder. *Fractals, Chaos, Power Laws: Minutes from an Infinite Paradise*. W. H. Freeman, 1992.

[60] M. Skurichina and R. Duin. Bagging and the random subspace method for redundant feature spaces. In *Proceedings of the Second International Workshop on Multiple Classifier Systems*, 2001.

[61] A. Strehl and J. Ghosh. Cluster ensembles -a knowledge reuse framework for combining multiple partitions. *Journal on Machine Learning Research (JMLR)*, 3, 2002.

[62] P. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison Wesley, first edition, 2005.

[63] A. Topchy, A. Jain, and W. Punch. Clustering ensembles: Models of consensus and weak partitions. *Pattern Aanalysis and Machine Intelligence*, 27(12):1866–1881.

[64] A. Topchy, A. Jain, and W. Punch. Combining multiple weak clusterings. In *Proceedings of the Third IEEE International Conference on Data Mining*, 2003.

[65] A. Topchy, A. Jain, and W. Punch. A mixture model for clustering ensembles. In *Proceedings of SIAM International Conference on Data Mining*, 2004.

[66] K. Wagstaff, C. Cardie, S. Rogers, and S. Schrödl. Constrained k-means clustering with background knowledge. In *Proceedings of the Eighteenth International Conference on Machine Learning*, 2001.

[67] E. Xing, A. Ng, M. Jordan, and S. Russell. Distance metric learning, with application to clustering with side-information. In *Proceedings of Advances in Neural Information Processing Systems*, 2003.

[68] H. Zengyou, X. Xiaofei, and D. Shengchun. Squeezer: an efficient algorithm for clustering categorical data. *Journal of Computer Science and Technology*, 17(5), 2002.

[69] Y. Zhou and S. Goldman. Democratic co-learning. In *Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence*, 2004.

[70] Z. Zhou and M Li. Tri-training: Exploiting unlabeled data using three classifiers. *IEEE Transactions on Knowledge and Data Engineering*, 17(11), 2005.

[71] X. Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005.

Curriculum Vitae

Muna Al-Razgan is a Saudi Arabian citizen. She received her Bachelor of Science in Mathematics from College of Education for Girls, in Riyadh Saudi Arabia. She obtained her Master of Science in Information Systems from George Mason University. She is currently working as consultant in Data Mining and Data Analysis at the Internal Audit Department (IAD) at the World Bank.