Informatik

Ensemble and Constrained Clustering with Applications

Inaugural-Dissertation
zur Erlangung des Doktorgrades
der Naturwissenschaften im Fachbereich
Mathematik und Informatik
der Mathematisch-Naturwissenschaftlichen Fakultät
der Westfälischen Wilhelms-Universität Münster

vorgelegt von
Daniel Duarte Abdala
aus São Paulo, Brazilien
- 2010 -

Dekanin/Dekan:	Prof. Dr. Matthias Löwe
Erster Gutachter:	Prof. Dr. Xiaoyi Jiang
Zweiter Gutachter:	Prof. Dr. Horst Bunke
Tag der mündlichen Prüfung(en):	16.12.2010
Tag der Promotion:	....

Abstract

The main focus of this thesis concerns the further developments in the areas of ensemble and constrained clustering. The goal of the proposed methods is to address clustering problems, in which the optimal clustering method is unknown. Additionally, by means of pairwise linkage constraints, it is possible to aggregate extra information to the clustering framework.

Part I investigates the concept of ensemble clustering. It presents a comprehensive review of the state of the art in ensemble clustering. It follows by discussing the impact of the ensemble variability in the final consensual result. Visualization of ensemble variability based on multidimensional scaling is also a topic addressed in this part. A software which is able to perform ensemble clustering using various existing consensus functions is also introduced. A consensus function based on random walker originally developed for image segmentation combination is adapted to the ensemble clustering problem. A lower bound is proposed to explore how well cluster ensemble methods perform in an absolute sense, without the usage of ground-truth. Finally, a study evaluating how well the general ensemble clustering techniques perform in the context of image segmentation combination closes this part.

Part II introduces an ensemble clustering method based on a new formulation for the median partition problem. The performance of this method is assessed in relation to other well known ensemble clustering methods.

Part III addresses the potential of ensemble techniques in the framework of constrained clustering. It presents a comprehensive review of the state of the art in constrained clustering and discusses the impact of considering constraints locally or globally. An experiment is presented comparing both approaches. A new clustering method is introduced combining both ensemble and constrained clustering. Constraints are introduced into three consensus functions. This part closes with an experimental evaluation, in which constraints are considered in different steps of the clustering ensemble framework.

In Part IV a review of the imaging protocol known as diffusion tensor imaging is presented, and a new fiber segmentation methodology based on the definition of pairwise linkage constraints is proposed to drive the semi-supervised segmentation process.

Acknowledgements

First and foremost, I would like to thank my adviser, Prof. Dr. Xiaoyi Jiang, for his continuous support and wonderful advisements regarding my work. I could never be more blessed, and it was truly an honor and a life experience to work with him. Professor Jiang, as I usually call him, was also very supportive in other aspects of my life here in Germany, being very understandable when I had to take a short leave for family reasons, and for that, I will always be thankful to him.

I remember well when I first read one of his papers and have thought: ”wow, this is a clear work presentation. I also would like to be able to do my research this way!”. In fact, Prof. Jiang thought me many things regarding the academic life, ranging from how to clearly address an idea to scientific events organization. I also would like to thank him for the opportunity of working in the IJPRAI. This two years experience gave me a clear insight of how a scientific journal works.

I would like to express my sincere thanks to the CNPq - Conselho Nacional de Pesquisa. My research would never be possible without the continuous support provided by this Institution. I'm honored to be one of the researchers chosen to be founded (process number 290101-2006-9), and I hope to be able to contribute to the research Brazilian community in the years to follow.

I am also very grateful to the DAAD (Deustche Akademischer Austausch Dienst) for proportionating me a 6 month German language course at the Goethe-Institut (Göttigen). Learning German properly made all the difference in my daily life in Germany.

I would like to express my sincere thanks to Prof. Cláudia Maria de Oliveira Souza and Prof. Joel Abdala (my dad) for the grammar correction. I wish I had eyes like your for finding typos. Thanks for teaching me that ”where” is not a ”wildcard”.

I will never be able to thank enough Mrs. Hildegard Brundestering for all the times she solved my bureaucratic problems here in Germany. The help she dispensed to me in finding a home, all the advisements regarding the necessary documentation to settle up in a city, health insurance, were a life saver in my first days here. She also saved me the trouble of showing up in the lab during holidays, always reminding me one day before: "Herr Abdala, morgen haben wir Feiertag.". The examples of her helpfulness could fill up this entire thesis, so I will simply say: "Frau Brunstering, thank you a lot for everything".

To all my colleagues in the Computer Vision and Pattern Recognition Group, my sincere thanks. It was a rich experience to work with you all. In special, I would like to thank my closest colleagues, namely Pakaket Wattuya, Kai Rothaus and Lucas Franek. To develop research with you guys was truly an awesome experience.

For my family, you all know I will never be able to thank you enough. Papa, despite the fact you almost scared me to death in my third year here in Germany during your stroke, please keep in mind that I truly believe no living being could ever be more fortunate than me by having you as my dad. Mom, you took my peace with all the advices about the various medicines I should take for all possible and imaginable diseases. To keep up with you recommendation, it was almost harder than to do my research itself, but again, you are truly a mom, I couldn't wish a better one. To my siblings Rachel, Déborah and Joel, since we were kids you stole my chocolate and have hidden countless Lego pieces driving me insane. However, I know better, the worst of you is the best any brother could ever hope for. You guys are awesome. Nona, I know you know how much a love you. You are a dream grandmother. Thank you a lot for the calls, for the money for shoes that I regret to assume now I spent in chocolate and foremost, for all your love.

I would like to thank Angela, my fiancée and soon to be wife. When I first came to Germany and you stayed in Brazil to attend to your second college course, I had feared we would not be able to endure this long separation. Fortunately, the separation was in fact simply geographic, since we almost daily talked to each other. The feeling of meeting you again after one year or so, with no awkwardness between us, was all the answer I needed to truly know you are the woman of my life.

Finally, I would like to thank God. Thank you for the opportunity of living this fulfilling experience my life has been.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Since some time, unsupervised clustering [77, 116, 170] methods have found their way from pure academic research to real life applications. Examples can be found in virtually any field relying on data analysis. In fact, it is one of the most common tools in this area. In the field of search results grouping [109, 139, 142, 175] clustering may be used to create a more relevant set of search results compared to keyword based search. An interesting example of such application is the Clusty [150] search engine. In social network analysis [98, 102, 117], clustering is also useful to identify informal communities not explicitly defined by its participants. It also can be used in epidemiological modeling of the spread of contagious diseases [46, 93]. In market research, clustering techniques are widely used by market planners. A common application is the partition of surveyed populations of consumers into market segments [75]. It can also be used to uncover relationships between different groups of customers and potential customers [95]. In image segmentation, clustering techniques are extensively used to partition image pixels in subsets representing objects or patterns. Medical [123], biological [5], and histological [113] are examples of images commonly analyzed using cluster based image segmentation techniques.

Most of the examples listed above use clustering algorithms specifically tuned to deal with the nuances inherent of the data. In fact, the clustering problem can be defined as stated by Kleinberg [91]: *"given an underlying set of points, partition them into a collection of clusters so that points in the same cluster are close together, while points in different clusters are far apart"*. There are few works about clustering independently of any particular algorithm. Kleinberg proved via an impossibility theorem that it is not possible for a clustering algorithm to satisfy at the same time the following three properties:

- **Scale-Invariance:** this property introduces the requirement that the cluster algorithm must not be sensitive to changes in the units of distance measurement;

- **Richness:** this property requires that all possible partitions in the data space should be reached provided a suitable distance function;

- **Consistency:** the third property states that a given partition is obtained using a distance function d. If distances between objects inside the clusters are reduced and the distances between clusters are enlarged, it should be possible to come up with a new distance function d' able to produce the same partition achieved by d.

Therefore, it is equally hard to design a clustering algorithm that could be applied to any kind of underlying data structure. A classical example of this difficulty can be found regarding the well known K-Means algorithm [110]. It is known to perform remarkably well over datasets showing clusters with hyper-spherical shapes, but it presents poorer results otherwise. Another example refers to hierarchical clustering. This family of cluster algorithms is known to work well on continuous clusters, not mattering the shape, but it is sensitive to noise and clusters proximity.

During the last years, two new promising clustering techniques emerged. Constrained clustering [15] takes advantage of known information about the dataset to aid the clustering process. Partially labeled data, expected maximum and minimum cluster size, and pairwise linkage pattern relationships are examples of information used by a constrained clustering algorithm. By means of extra information about the data, better and more reliable clustering results can be obtained. Ensemble clustering [7, 57, 145], on the other hand, combines multiple partitions into a single solution, aiming to produce a smoother result, and possibly, even improving the final result compared to any individual partition in the ensemble. By means of those new clustering methods, a broad new range of applications can be addressed.

The main focus of this thesis concerns further developments in the areas of ensemble and constrained clustering. The goal of the proposed methods is to address clustering problems, in which the optimal clustering method is unknown. Additionally, by means of pairwise linkage constraints, it is possible to aggregate extra information to the clustering framework.

1.1 Motivation

Recently, two new clustering techniques were introduced. They are called ensemble and constrained clustering. Although very promising, both techniques are reasonably new, presenting room for further investigations. For one side, ensemble clustering is a viable option to address the problem of selecting a fitting clustering algorithm in cases where the dataset distribution is unknown. It also provides a way to deal with noisy data and outliers. At the time, a myriad of consensus functions exists with new ones being proposed regularly. The foundation in which this method is based, namely the ensemble of partitions, is commonly neglected. A considerable number of related papers rarely address the ensemble generation step, crucial for the obtainment of a good consensus partition at the end of the process. Aspects as ensemble variability and its impact on the final consensual partition are seldom cited. Constrained clustering is an extension of general clustering able to incorporate extra information to the clustering process. The most investigated type of constraints refers to pairwise linkage constraints, in which pairs of patterns are deemed to be in the same or in different clusters. The performance increase in comparison to unconstrained clustering methods tends to be proportional to the quantity of side information provided. Furthermore, most works in the field of constrained clustering present mainly synthetic evaluation data, in which constraints are automatically generated based on a known ground-truth.

The main motivation of this thesis is to propose a unification approach considering both ensemble and constrained clustering into a single framework. Additionally, a constrained clustering algorithm is proposed to address the problem of fiber segmentation. The remainder of this section gives a brief overview of the fiber segmentation problem.

Recently, a new magnetic resonance imaging protocol called diffusion tensor imaging (MR-DTI) [18] was introduced. By series of post-processing steps, it allows the creation of a three-dimensional representation of living fibrous tissues, e.g. the human brain. The task of processing a MR-DTI series into a collection of brain fibers is called fiber tracking. In fact, the advent of fiber tracking alone, from a simple visualization point of view is a considerable contribution. Progressively, new neuro-anatomical material in the form of 3D atlases is being made available, in general for educational purposes. A good example was introduced in [159]. No longer specialists are required to rely on projections over 2D images, or artistically created representations.

Fiber tracking shows a very detailed representation of the white matter, allowing

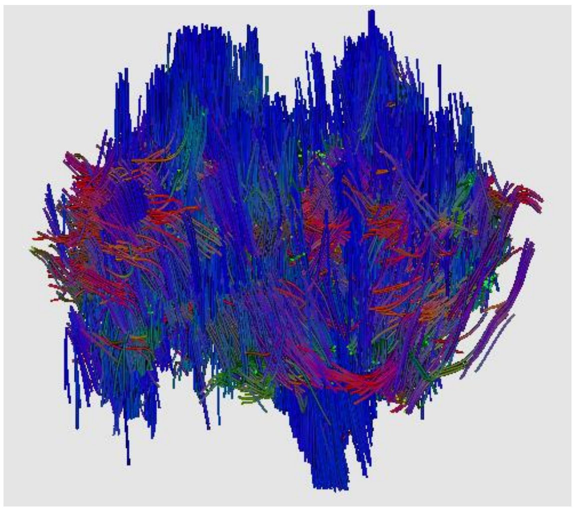


Figure 1.1. Example of fiber segmentation

differentiation on its inner structures (hard to be isolated in anatomical imaging such as MR-T1/T2 or CAT-scans). It also allows the identification of connections between different functional brain regions. This can be done by isolating subsets of fibers also called fiber bundles, connecting two or more functional regions.

Fiber tracking maps can be tricky to visualize and to interpret (see Figure 1.1). Given its three-dimensional nature and the huge amount of fibers (potentially tens of thousands of fibers) it becomes hard to extract only by visual inspection useful information in order to aid medical diagnosis. A new task called fiber segmentation was introduced [11]. Its objective is to assign meaning to the potentially incomprehensive set of fibers. Although named fiber segmentation, the process has little to do with segmentation in the sense it is given within the context of image segmentation. In fact, the process can be better seen as the partition of the set of fibers (objects) into meaningful subsets, what can be seen as a clustering task.

Usually, medical institutions with access to fiber tracking technology using it in a daily clinical basis rely heavily on manual segmentation. The reasons for this choice of tool can be summarized as follows:

- The user has complete control of the segmenting process;
- It is easier to account to anatomical changes induced by the disease being studied, resulting from chronic diseases, natural malformations, or anatomical abnormalities;
- The user tends to feel more confident in relying on a result of which he has complete control over the process of obtaining it.

The second motivational objective of this thesis is to propose a more reliable way to perform semi-supervised fiber segmentation. Additionally, the proposed method should satisfy the list of reasons for choosing manual segmentation given above. A fiber segmentation procedure is devised that takes advantage of the new methodologies in clustering analysis cited earlier in this section, namely constrained clustering. In order to achieve this objective, a new constrained clustering algorithm had to be proposed. It requires as input the user to identify pairwise relationships between fibers or set of fibers as belonging to the same group or fibers close to each other belong to different groups. The segmentation can be refined interactively until an acceptable result is reached.

1.2 Objectives

The main objectives of this thesis are summarized as follows:

- To introduce a new consensus function for ensemble clustering entitled sum of pairwise distances. It is based on a new formulation of the median partition problem;
- To consider the concepts of ensemble and constrained clustering altogether and to propose a suitable algorithm able to take advantage of both methods;
- To propose an algorithm to segment fiber tracking of the human brain into meaningful substructures demanding less effort from the specialized user.

Along with the main objectives, there are also relevant topics investigated in this thesis. They are related to the main objects above described.

- To investigate the problem of ensemble variability and its impact in the consensus partition obtained via consensus functions;
- To propose a viable way to measure ensemble variability by means of a variability index;
- To propose a visualization scheme able to detect variability issues within an clustering ensemble;
- To investigate the applicability of the random walker image segmentation algorithm to the context of ensemble clustering;

- To evaluate the usefulness of a lower bound designed for the generalized median problem to ensemble clustering;
- To investigate the viability of applying existing ensemble clustering methods to address the problem of image segmentation combination.

1.3 Thesis Organization

The remainder of the thesis is organized as follows:

Chapter 2 presents an overview of some fundamental concepts and algorithms that are required for understanding the work presented in this thesis. These topics are used throughout the thesis, such as the disambiguation of general terms widely used in the text, the mathematical notations adopted by the equations and algorithms, the ensemble generation schemes, twelve commonly used evaluation measures, a mathematical formulation for the median partition problem, and the description of the four used databases.

Part I - Ensemble Clustering comprehends Chapters 3, 4, 5, and 6.

Chapter 3 presents a comprehensive review of ensemble clustering. The two steps of ensemble clustering, namely, ensemble generation and consensus functions are discussed in detail. The variability of ensembles achieved by different ensemble generation schemes is discussed and means to measure and visualize variability are proposed. A taxonomy of the existing consensus functions is presented. The existing ways to evaluate the accuracy of results produced by ensemble clustering methods are discussed in detail. This chapter ends with the presentation of a software developed to simplify the usage of ensemble clustering consensus functions.

Chapter 4 presents the adaptation of a random walker consensus function to work with ensemble clustering problems. The original method is reviewed and the need of an alternative graph-based representation to deal with datasets other than images is identified and proposed. This chapter ends with an experiment comparing the results obtained by the random walker consensus function to other well known ensemble clustering methods.

Chapter 5 presents a lower bound to explore how well cluster ensemble methods perform in an absolute sense without the usage of ground-truth. The chapter follows by presenting other two lower bounds that can also be used for ensemble clustering. The lower bound proposed is evaluated for the cases of weighted and unweighted ensemble clustering.

Chapter 6 explores the idea of consensus clustering to address the problem of image segmentation combination. The framework proposed for image segmentation combination based on general consensus functions is presented, followed by experimental results obtained using the Berkeley image database as evaluation basis.

Part II - A New Consensus Function for Ensemble Clustering comprehends Chapter 7

Chapter 7 introduces a new formulation to address the problem of finding the median of objects. The problem is motivated and the proposed formulation is mathematically defined. The implementation details for this new method to compute the median of objects in the context of ensemble clustering is given, followed by the introduction of a cluster validity index based on it. This chapter ends with an experiment comparing the results obtained by this new ensemble clustering method to other well known ensemble clustering consensus functions.

Part III - Constrained Ensemble Clustering comprehends Chapters 8 and 9.

Chapter 8 reviews the constrained clustering topic. The existing types of constraints are described, followed by a discussion about the relevance of constraints with respect to possible gain in accuracy they can bring to the clustering process. The existing constraining methods are reviewed and used in conjunction with the existing types of constraints to create a constrained clustering taxonomy. This chapter ends with a quantitative evaluation comparing global and local constraining methods.

Chapter 9 addresses the potential of ensemble techniques in the framework of constrained clustering. The constrained ensemble clustering framework is presented and three constrained consensus functions are proposed. This chapter ends by presenting a quantitative study comparing the results of the constrained consensus functions proposed to standard ensemble and constrained clustering methods.

Part IV - Fiber Segmentation comprehends Chapters 10 and 11.

Chapter 10 reviews the fundamental concepts of diffusion tensor imaging. The mathematical foundation underlying the computation of diffusion tensors, the visualization schemes commonly applied on DTI, and the fiber tracking process are the topics reviewed in this chapter.

Chapter 11 addresses the problem of fiber segmentation. This chapter starts by reviewing the existing fiber segmentation methods. A new semi-supervised method to perform fiber segmentation using constrained clustering is proposed and experimental results are presented, in which a number of fiber structures is successfully segmented by means of the algorithm proposed.

The contributions of this thesis in summary and final conclusions are given in Chapter 12.

Chapter 2

Fundamentals

This chapter gathers fundamental concepts, algorithms and information about the databases that are used throughout the thesis. Section 1 provides a comprehensive disambiguation of the general terms used. Some of the terms belong to relatively new areas and therefore, little consensus among different authors exists. Choices are made to maintain the text's cohesion. The variants of such terms are hereby named in order to ease the reading. Section 2 gathers the mathematical notation used in order to serve as a reference point that can be revisited in cases of doubt during the inspection of specific sections of the text. However, any additional nomenclature needed is properly defined *in locus.* Section 3 describes the ensemble generation schemes as well as the algorithms used to create the partitions. This topic is located here since it plays a central role for the ensemble clustering methods reviewed or proposed in this thesis. Additionally, the ensemble generation schemes require a number of details to be specified. Section 4 gives a brief review of the existing evaluation measures, commonly used to assess the quality of clustering algorithms. Some of those measures are used during the evaluation step by different algorithms all around the thesis. Section 5 reviews the concept of computing median of objects. Median concept plays an important role in the consensus clustering methods. It is also used by the constrained ensemble clustering methods introduced in this thesis. Additionally it works as a basis for the sum of pairwise distance method proposed in Chapter 7. Finally, Section 6 presents a detailed description of the four databases used by the various experiments in this thesis.

2.1 General Terms and Disambiguation

Throughout this work, some terms and concepts are used extensively. In order to make easy the access of such terms in times when a reminder is needed, they are summarized in this section. The terms are organized in two main classes: a) clustering, and b) fiber segmentation. Some terms belong to recent theoretical proposals. Thus, during the time this thesis was written, little consensus regarding the nomenclature existed. For the sake of clarity, the interchangeably terms are here listed. In other cases, the terms can come from different areas using understandably different nomenclatures, but ultimately representing the same concept.

Clustering Related Terms

- Dataset - data source, or set of patterns provided to the algorithms;
- Pattern, Object, Instance - one single observation over a phenomena;
- Partition - subdivision of the patterns of a given dataset into meaningful groups;
- Partitioning, Clustering - the process by which partitions are created;
- Ground Truth - it is a human made/inspected partition of the dataset assumed to be the ”correct” answer. It is commonly used to assess the performance of clustering algorithms by means of direct comparison;
- Similarity, Distance Measure - it is a function that computes the similarity between two partitions;
- Consensus Clustering, Clustering Combination, Ensemble Clustering - it is two-step process that generates an ensemble of partitions over the same dataset and combines them into a single consensual result;
- Cluster Ensemble (CE) - it is defined as a collection of partitions of the same dataset;
- Consensus Function (CF) - it is a process that combines a cluster ensemble into a final consensual partition;
- Consensus Partition (CP) - it is defined as the output of a consensus function;
- Constrained Clustering (CC) - it is a method that incorporates side information into general clustering methods;

- Constraint - it is defined as a limitation about how the data can be clustered.

Fiber Segmentation Related Terms

- Fiber, Curve, Polyline - it is a single object composed by a number of points, in this context a representation of a white matter's neural fiber;
- Fiber Tracking (FT) - it is the result of processing a diffusion tensor magnetic resonance imaging series into a set of space-curves. It can be understood as a collection of polylines representing the connectivity of functional regions of the brain;
- Fiber Bundle (FB) - it is defined as a subset of fibers from a fiber tracking;
- Fiber Segmentation (FS) - it is the process of partitioning the fiber tracking into various meaningful fiber bundles;
- DWI and DTI - DWI stands for Diffusion Weighted Imaging and DTI to Diffusion Tensor Imaging. DWI is the actually MRI protocol but DTI is popular, since the computation of the diffusion tensor's volume is the most common post-processing, used by virtually any practical DWI application.

2.2 Mathematical Notation

X	dataset
N	stands for the number of patterns or objects in a set
x, y, p, q	pattern
C	a subset of patterns of X grouped together and sharing the same label
K	number of clusters
P, Q, F, P_i	partition of a dataset into K clusters
$d(\cdot, \cdot)$	similarity measure
$\mathbb{P}$	ensemble of partitions
M	number of partitions in $\mathbb{P}$
CM	co-association matrix
$m_{i,j}$	number of times patterns i and j share the same label in $\mathbb{P}$
Γ	lower bound
$\sim$	binary relation "similar", it is used to represent a must-link
$\nsim$	binary relation "dissimilar", it is used to represent a cannot-link
$\mathcal{ML}$	it is the set of must-link constraints
$\mathcal{CL}$	it is the set of cannot-link constraints

The various algorithms and equations presented in this thesis use a regular notation throughout the entire text. Therefore, in cases of doubts regarding the notation used when inspecting a given algorithm or equation, the reader should refer to this section. Any additional notation required is defined *in locus*, and properly explained.

2.3 Ensemble Generation Schemes

One of the most important aspects of the dataset's pre-processing on this thesis refers to the generation of ensemble of partitions. In order to systematically evaluate the ensemble clustering, and the consensus functions proposed, a series of ensemble generation schemes are proposed. This section describes each one of them, correlating them with a fix nomenclature. The generation schemes are detailed later on in Chapter 3. For each ensemble scheme, ten ensembles using the same strategy are generated in order to improve the noise resistance.

The first generation scheme refers to the selection subsets of attributes. Four ensembles for each dataset are generated using this scheme. The nomenclature adopted is the following, **KMclicksXX-YY**. "KM" stands to K-Means algorithm. The number of clusters is extracted from the ground-truth. "clicks" means that a subset of the attributes is used, "XX" is the number of partitions in the ensemble and "YY" is the percentage of attributes used. Sixteen ensembles are generated this way, as listed on Table 2.1.

Table 2.1. List of ensembles generated using K-Means and subsets of attributes

KMclicks10-03	KMclicks10-05	KMclicks10-07	KMclicks10-09
KMclicks20-03	KMclicks20-05	KMclicks20-07	KMclicks20-09
KMclicks30-03	KMclicks30-05	KMclicks30-07	KMclicks30-09
KMclicks40-03	KMclicks40-05	KMclicks40-07	KMclicks40-09

The second generation scheme uses both K-Means with a random number of target clusters and subsets of attributes. It follows the same system described for the previous case. The number of target clusters is picked at random for each partition within the range $[K, 2K]$. The nomenclature adopted is the following, **KMclicksrandKXX-YY**. Sixteen ensembles are generated this way. Their names are listed in Table 2.2.

The third generation scheme uses K-Means as clustering algorithm and a random number of target clusters. Four ensembles are generated with numbers of partitions 10, 20, 30 and 40, respectively. The nomenclature adopted is **KMrandXX**.

Table 2.2. List of ensembles generated using K-Means with random K and subsets of attributes

KMclicksrandK10-03	KMclicksrandK10-05	KMclicksrandK10-07	KMclicksrandK10-09
KMclicksrandK20-03	KMclicksrandK20-05	KMclicksrandK20-07	KMclicksrandK20-09
KMclicksrandK30-03	KMclicksrandK30-05	KMclicksrandK30-07	KMclicksrandK30-09
KMclicksrandK40-03	KMclicksrandK40-05	KMclicksrandK40-07	KMclicksrandK40-09

The fourth generation scheme also uses K-Means as clustering algorithm. It simply runs the algorithm with the number of target clusters extracted from the ground-truth. The nomenclature adopted is **KMXX**.

The fifth generation scheme is produced using K-Means with number of target clusters is picked at random for each partition within the range $[K, 2K]$. The nomenclature adopted is **KMrandXX**.

The final generation scheme uses a number of different algorithms to produce the ensemble. The nomenclature adopted is **diffAlgs**. The ensembles of partitions are generated using different ensemble generation strategies as described earlier. Among one of the generation strategies, it is required different clustering algorithms to be used. The list bellow refers to the collection of algorithms used in order to generate the ensembles.

- **Mean shift algorithm** [61] - The idea behind mean shift is to consider the points in the feature space as an empirical probability density function. Dense regions in the feature space correspond to the local maxima. For each data point, it performs a gradient ascendent procedure on the local estimated density until convergence is reached. The stationary points of this procedure represent the modes of the distribution.

- **Kernel K-Means** [31] - Before clustering, points are mapped to a higher-dimensional feature space using a nonlinear function. Subsequently, kernel K-Means partitions the points by linear separators in the new space.

- **Adaptive Affinity Propagation** [59] - It takes as input measures of similarity between pairs of data points. Real-valued messages are exchanged between data points until a high-quality set of exemplars and corresponding clusters gradually emerges.

- **Mixture Model** [53] - Clustering methods based on mixture models represents mathematically each cluster by a parametric distribution. The Gaussian and Poisson's distributions are commonly used. The entire data is modeled by a mixture of these distributions. An expectation maximization algorithm is used to systematically stabilize the distributions. The one with highest probability is selected as the clustering result.

- **K-Means** [110] - This is perhaps the most popular clustering algorithm. It seeks an optimal partition of the data by minimizing the sum of square error criterion which is an iterative optimization procedure.
- **Hierarchical** [170] - Hierarchical clustering is also a popular algorithm. It seeks to systematically agglomerate patterns based on its pairwise distance.
- **Spectral Clustering** [32] - The basic idea is to construct a weighted graph from the initial data set where each node represents a pattern. Each weighted edge takes into account the similarity between two patterns. This method models clustering as a graph cut problem, which can be tackled by means of the spectral graph theory. The core of this theory is the eigenvalue decomposition of the Laplacian matrix of the weighted graph obtained from the data.

2.4 Measures for Comparing Partitions

The objective of a similarity function is to provide a measure of how similar/dissimilar two given partitions are. They are among the fundamental concepts in this work, similarity measures figure among the most widely used.

Table 2.3. List of distance/similarity measures

Distance	Metric
Rand	No
Adjusted Rand	No
Jacard	No
Mirkin	Yes
F-measure	No
Folks&Mallows	No
Dongen	Yes
Bipartite Graph Matching	No
Mutual Information	No
Variation of Information	Yes
Error Rate	No

This subsection reviews some details of those measures. There are many different ways to measure the similarity or distance between two partitions. However they can be organized in the following classes:

a) Counting Pairs;

b) Set matching;

c) Information-Theoretically.

Table 2.3 lists the similarity measures. Additionally, it also indicates if they are metrics. Metric distances are valuable since they make the criterion more understandable and match the human intuition better than an arbitrary distance function. In order for a distance function to be a metric they must obey the four given conditions:

1. $d(x, y) \geq 0$ (non-negativity)
2. $d(x, y) = 0$ if and only if $x = y$ (identity of indiscernible)
3. $d(x, y) = d(y, x)$ (symmetry)
4. $d(x, z) \leq d(x, y) + d(y, z)$ (sub-additive / triangle inequality)

2.4.1 Pair Counting Based Comparison Methods

There is a number of indexes possible of being computed using four simple counting variables. They are defined as follows. Given two partitions P and P' of a set X of N patterns, all pairs of patterns $(x_i, x_j), i \neq j$ from $X \times X$ are considered. There are four possible situations where those pairs could be accommodated:

- N_{11} - number of pairs of patterns in the same cluster in both P and P';
- N_{00} - number of pairs of patterns in different clusters in P and P';
- N_{10} - number of pairs of patterns in the same cluster in P but not in P';
- N_{01} - number of pairs of patterns in the same cluster in P' but not in P.

The four counts always satisfy the equality:

$$N_{11} + N_{00} + N_{01} + N_{10} = \frac{N(N-1)}{2} \tag{2.1}$$

Several distance measures are based on these four counts. The definition of the most relevant is presented in the paragraphs that follow.

Rand Index: The rand index [128] is a similarity measure that allows the evaluation of clustering algorithms. It is done by comparing two partitions being one a known ground-truth. It is defined as:

$$Rand(P_1, P_2) = \frac{N_{11} + N_{00}}{(N(N-1))/2} \tag{2.2}$$

It gives a measure of similarity within the range $[0, 1]$ The value 0 is produced in cases that the two partitions being compared are completely different. It is important to notice that the rand index is not corrected by chance as shown by Hubert and Arabie [76]. In this same work, Hubert proposed a new version of this index called, adjusted rand index, accounting to this limitation. It is defined as follow:

$$ARand(P_1, P_2) = \frac{Rand(P_1, P_2) - E[Rand]}{1 - E[Rand]} \tag{2.3}$$

The adjusted rand index has also a wider range $[-1, 1]$, where 1 is obtained when the two partitions are identical.

Jacard Index: The Jacard Index [16] gives a similarity measure within the range $[0, 1]$. It is defined as follows:

$$\mathcal{J}(P_1, P_2) = \frac{N_{11}}{N_{11} + N_{10} + N_{01}} \tag{2.4}$$

Mirkin Distance: The Mirkin distance [116] is still another adjusted version of rand index. It is important to notice that the Mirkin distance is in fact a metric. It gives 0 if the two partitions are identical and a positive value otherwise. It is defined as follows:

$$\mathcal{M}(P_1, P_2) = 2(N_{10} + N_{01}) \tag{2.5}$$

F-Measure: Finally, the F-measure [14], based on the precision and recall measures, can also be computed using the four counting variables described earlier. It is defined as follows:

$$Precision(P_1, P_2) = \frac{N_{11}}{N_{10}} \qquad Recall(P_1, P_2) = \frac{N_{11}}{N_{01}} \tag{2.6}$$

$$FMeasure(P_1, P_2) = \frac{Precision \times Recall}{Precision + Recall} \tag{2.7}$$

Fowlkes & Mallows: Folks and Mallows [52] introduce the following index:

$$\mathcal{F}(P_1, P_2) = -1\sqrt{W_1(P_1, P_2) \times W_2(P_1, P_2)} \tag{2.8}$$

The W_1 and W_2 values are computed as showed bellow. This index also returns a value within the range $[0, 1]$.

$$W_1(P_1, P_2) = \frac{N_{11}}{\sum_{i=1}^{k} N_i \times (N_i - 1)/2} \qquad W_2(P_1, P_2) = \frac{N_{11}}{\sum_{j=1}^{l} N_j \times (N_j - 1)/2} \tag{2.9}$$

where N_i stands for the size of the *ith* element in C_1 and N_j the *jth* element in C_2. The terms W_1 and W_2 represent the probability that a pair of patterns, which are in the same cluster under C_1 are also in the same cluster under C_2, and vice versa.

2.4.2 Set Matching Methods

Error Ratio: The error ratio is a direct measure that gives the percentage of patterns wrongly classified compared to a known ground-truth. It is computed by matching the ground-truth information available with the partition to be compared. The correspondence between the two sets is established by computing all possible label permutations and then retaining the maximum. An efficient way to compute the label permutations is achieved by means of the Hungarian algorithm [55].

Dongen Index: The Dongen Index [43] is a distance based on the matching of sets. It takes the maximum value only if the two partitions are exactly the same. The term $a(P_1, P_2)$ is defined as follows:

$$a(P_1, P_2) = \sum_{C_i \in P_1} \max_{C_j \in P_2} |P_i \cap P_j| \, (N_j - 1)/2 \tag{2.10}$$

The Dongen index is defined in Equation (2.11). It was also proven that this distance is a metric.

$$\mathcal{D}(P_1, P_2) = 2 \times N - a(P_1, P_2) - a(P_2, P_1) \tag{2.11}$$

Bipartite Graph Matching: The $\mathcal{BGM}$ index [80] computes a one-to-one correlation between image element clusters, trying to maximize their relationship. There are two partitions: C_1, representing the segmentation, and C_2, representing the ground-truth. It considers each cluster of the C_1 and C_2 as vertices of a bipartite graph. Edges are added between each vertex of the two partitions and they are valued as $|c_{1i} \cap c_{2j}|$, a value that can be directly extracted from the matching matrix. The maximum-weight bipartite graph is then defined as the subgraph $\{(c_{1i1}, c_{2j1}), \cdots, (c_{1ir}, c_{2jr})\}$ where only the edges from c_{1i} to c_{2j} with maximum weight are present. The $\mathcal{BGM}$ index is defined as follows:

$$\mathcal{BGM}(C_1, C_2) = 1 - \frac{\sum_{i=1}^{I} \sum_{j=1}^{J} \max(v_{i,j}))}{n} \tag{2.12}$$

where $v_{i,j}$ are the edge weights, i is the number of clusters in C_1 and j is the number of clusters in C_2. The total number of elements is given by n. The $max(v_{i,j})$ term guarantees that only the maximum weight value for each partite connection should be computed in the final weight summation.

2.4.3 Information Theoretic Methods

This last class of comparison methods is based on the mutual information. This is a well known concept in information theory that measures how much information a random variable X is obtained by observing the random variable Y.

Mutual Information: MI [141] is a widely used index. It measures how much information is shared between the two inspected variables. It is defined as follows:

$$MI(P_1, P_2) = \sum_{(C_{P_1}, C_{P_2})} p(C_{P_1}, C_{P_2}) \log \frac{p(C_{P_1}, C_{P_2})}{p(C_{P_1}) p(C_{P_2})} \tag{2.13}$$

where $p(C_{P_1}, C_{P_2})$ is the joint distribution and $p(C_{P_1})$ and $p(C_{P_2})$ the marginal probability functions.

A normalized version of the mutual information can be computed as follows:

$$NMI(P_1, P_2) = 1 - \frac{1}{\log(k.l)} \sum_{C_{P_1} \in P_1} \sum_{C_{P_2} \in P_2} p(C_{P_1}, C_{P_2}) \log \frac{p(C_{P_1}, C_{P_2})}{p(C_{P_1})p(C_{P_2})} \tag{2.14}$$

Variation of information: VI [114] is still another information theoretic comparison method. It measures how much information is gained or lost in changing the random variable P_1 into P_2.

$$VI(P_1, P_2) = H(P_1) + H(P_2) - 2MI(P_1, P_2) \tag{2.15}$$

where $H(P)$ represents the entropy of the set P and can be computed as follows:

$$H(P) = -\sum C_p \in Pp(C_p) \log(C_p) \tag{2.16}$$

2.5 Median of Objects

The concept of averaging a set of objects to produce a single representative of the whole set was already extensively investigated given its usefulness in various areas of science, such as engineering and economy. Averaging provides clear advantages as such the possibility of having a single representative of a collection to be considered.

It is easy to compute median of numbers by means of statistical methods. However, the concept of averaging complex objects such as graphs or images requires a more expert definition. One powerful tool for this purpose is provided by the *generalized median* concept. It is defined as follows.

Given a set S of objects existing in a feature space U and a distance function $d(p, q)$ defined between any given objects $p, q \in U$, find the object $\bar{p}$ which minimizes the sum of distances between all objects in U. Equation (2.17) given the formula or generalized median.

$$\bar{p} = \arg\min_{p \in U} \sum_{q \in S} d(p, q) \tag{2.17}$$

Intuitively, the concept is very simple. All possible objects in U are considered as possible solutions to the problem. The object presenting the minimum distance to all objects in S is the optimal median. Unfortunately, it is easy to see that this problem is computationally intractable. In fact it was proven [10] that the median partition problem is $\mathcal{NP}$-complete for many reasonable distance functions.

A related concept to the generalized median known as *set median*, It only considers the elements existing in S as possible solutions. The set median is mathematically defined as follows:

$$p^* = \arg\min_{p \in S} \sum_{q \in S} d(p, q) \tag{2.18}$$

The set median may serve as an approximate solution for the generalized median. Note that neither the generalized median nor the set median is unique. In practice, suboptimal approaches [105, 140] are applied to solve the optimization problem. In this thesis, the concept of median of objects is extensively used in the various algorithms presented regarding the problem of ensemble clustering.

2.6 Databases

Throughout this work, different databases are used. This section summarizes all of them, giving some in-depth information.

Synthetic Datasets

In the experiments some synthetic datasets are used in order to evaluate the algorithm's behavior in well known situations. The synthetic datasets are presented in Table 2.4.

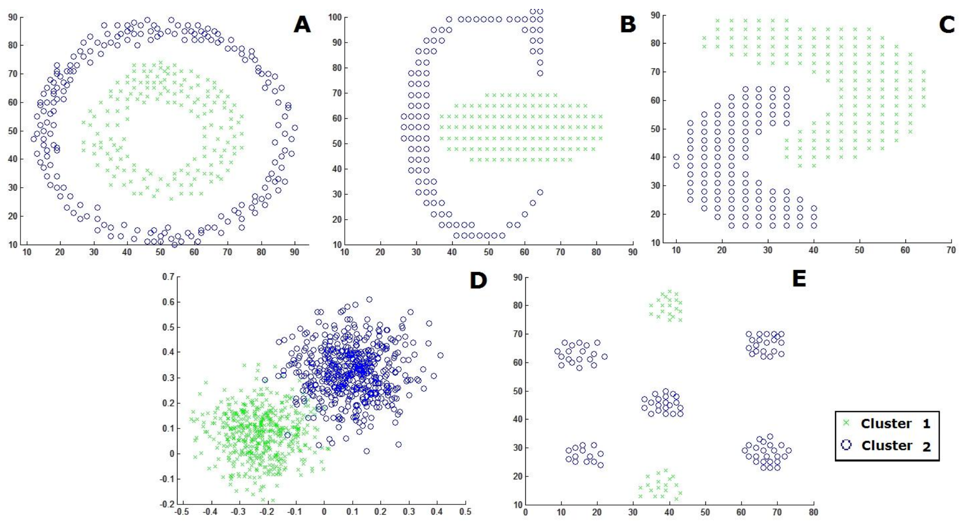


Figure 2.1. Representation of the 2D synthetic datasets

Half-rings is a classical problem. This dataset is composed of 269 patterns distributed in 2 classes (160-class 1; 109-class 2). The two-rings dataset is known to be a hard problem for hyperspherical algorithms such as K-Means. The dataset has 326 patterns (165- class 1; 161-class 2). C-ellipse is a dataset in which a C-shaped cluster embraces another elliptic cluster. It contains 225 patterns (85-class 1; 140-class 2). The scattered dataset contains 132 patterns (38-class 1; 94-class 2). The last two artificial data sets from [140] are included into the experiments. The first dataset (2D2K) contains 500 2-D points from two Gaussian clusters and the second dataset (8D5K) contains 1000 points from five multivariate Gaussian distributions (200 points each) in 8D space.

Table 2.4. List of synthetic datasets used in the experiments

Dataset	N. patt.	N. attr.	N. clust.	distribution
C-Ellipsoid	225	2	2	85 - class 1; 140 - class 2
Half-rings	269	2	2	160 - class 1; 109 - class 2
Scatered	132	2	2	38 - class 1; 94 - class 2
Two-rings	326	2	2	165 - class 1; 161 - class 2
2D2K	1000	2	2	500 - class1; 500 - class 2
8D5K	1000	8	5	200 patterns on each class

Figure 2.1 is a graphic representation of the five 2-D datasets. (A) refers to two-rings, (B) C-Ellipse, (C) Half-rings, (D) 2D2K, (E) scattered datasets.The sixth datasets cannot be graphically represented due the excessive number of dimensions.

UCI Irvine Datasets

The UCI Irvine Machine Learning Repository [56] is a collection of datasets extensively used by the machine learning community for empirical experimentation in machine learning algorithms. Its usage ensures cross-evaluation among similar algorithms. This fact alone ensures its validity. It is comprised of circa 190 datasets organized by different criteria, such as types of analysis (categorical, regression, clustering and others), types of attributes and data type, among others. For all datasets, a single ground-truth is provided. In this work 25 datasets are selected given the fact many of the available datasets are not suitable for the algorithms here proposed. In some cases, the datasets have to be edited in order to correct for missing or non-numerical values. Table 2.5 summarizes the selected datasets as well as the number of patterns, the number of attributes, the number of clusters, and a marker indicating if the dataset had to be edited in order to account to non-numerical or missing attributes.

Table 2.5. Setected UCI datasets

Dataset	N. patters	N. attributes	N. clusters	edited
balance	625	4	3	No
breast	683	9	2	No
control	600	60	6	No
ecoli	336	7	8	No
glass	214	9	7	No
haberman	306	3	2	No
heart	270	13	2	No
ionosphere	351	34	2	No
iris	150	4	3	No
lung	27	56	2	Yes
mammo	830	5	2	Yes
optic	1000	64	10	Yes
parkinsons	195	22	2	No
post-op	87	8	3	Yes
protein	116	20	6	Yes
satellite	6435	36	7	No
sonar	208	60	2	No
soybean	47	36	2	No
spect	267	22	2	No
spectf	267	44	2	No
taeval	151	5	3	No
tic-tac-toe	958	9	2	No
transfusion	748	4	2	No
wine	178	13	3	No
yeast	1484	8	10	No

Berkeley Image Datasets

The Berkeley segmentation database [112] is a large collection of color images. It provides a series of human made segmentations from different sources, working as ground-truth. Each image is segmented by more than one expert, allowing a trustworthy source for empirical evaluation. It is comprised by three hundred images divided in two parts, a training set of 200 images, and a test set of 100 images. All images have the same size 481×321 pixels or its landscape version 321×481 pixels. This is one of the most popular image databases, being largely used in scientific publications. The main reason for this preference is regarding the fact that for each image a number of ground-truths produced by different human experts is available.

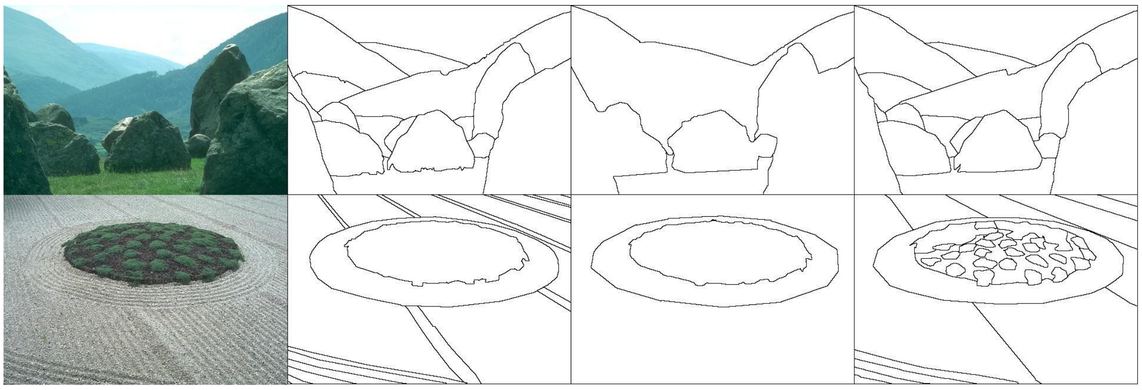


Figure 2.2. Examples of color images from the Berkeley image database

Figure 2.2 shows two examples of images and ground-truth extracted from the Berkeley image database. The original images are displayed in the left column followed three sample ground-truths for each image with different degree of details. Image 241004 (upper row), has ground-truths with 17, 6, and 18 segments. Image 86016 (lower row) has ground-truths with 24, 4, and 41 segments, respectively.

DTI Datasets

DWI datasets are generated by a Siemens Magneton Sonata $1.5T$, $TR = 9.75$, $TE = 4s$ with field of view set to 220 mm. The series comprise of 7 volumes (one un-weighted anatomical reference series, and 6 weighted ones encoding 6 principal diffusion directions) of 19 images each. The size of the acquired images is 128×128 pixels with voxel size $1,79 \times 1,79 \times 6,5$ mm. FT is generated using the MedINRIA software [148].

Figure 2.3 shows a section of DTI series produced by the equipment cited above. Note that the unweighted image is in fact a $T1$ image. However, the resolution is much smaller compared to standard $T1$ series (usually 512×512 pixels with

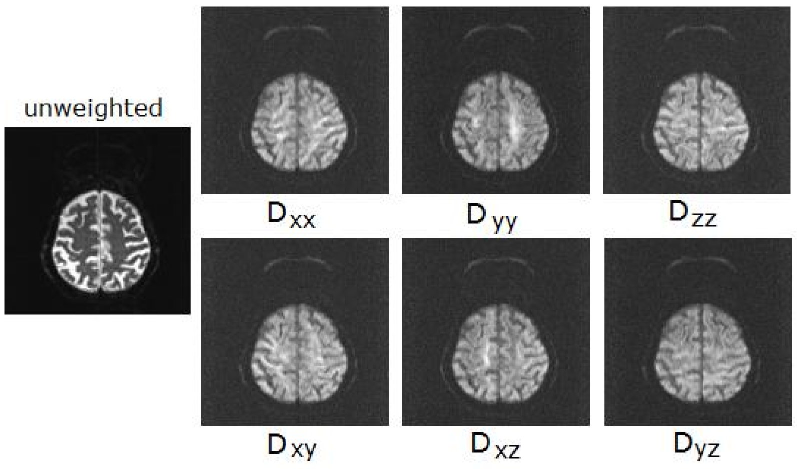


Figure 2.3. Example of one section of a DTI series

$1 \times 1 \times 1$ *mm* voxel volume and an average of 100 images per series). The weighted images showed in the right side of the picture, present a very poor visual resolution. This is due to the fact they encode the diffusion direction of water molecules with no regard to anatomical details.

For the purposes of this work, the scanned DTI series are not directly used. Instead, a post-processing called fiber tracking takes place. An example of fiber tracking produced over a DTI series is shown in Figure 2.4. This is a considerably complex processing step that is properly reviewed later on this thesis. Nevertheless, it is important to highlight that, for the purposes of this work, the object of interest are the fiber trackings and not the DTI itself.

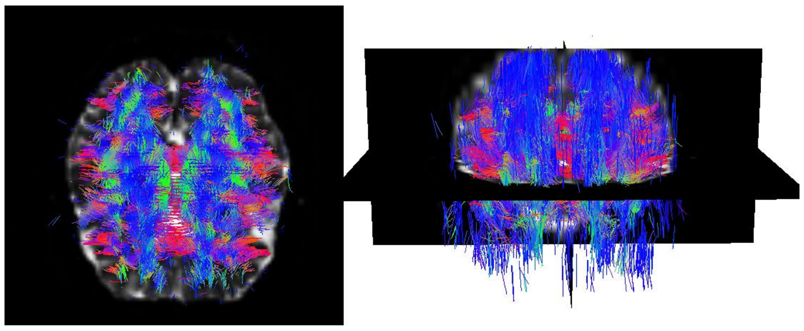


Figure 2.4. Two views of the same fiber tracking

Part I

Ensemble Clustering

Chapter 3

Ensemble Clustering

Clustering combination, also known as ensemble clustering, has emerged as a valid option in data clustering. It is an elegant way to deal with the problem of choosing the fittest clustering result in cases which little or nothing is known about the dataset. It also works as a way to smooth the final result when different partitions can potentially present dissimilar distributions. Finally, it is also a valid way to improve the final result. This is due to the fact it seeks to gather correct evidence among all the partitions merging it in a final consensual result. The fundamental ideas of ensemble clustering can be found in supervised learning [99]. It wasn't until the last decade the idea of combining the results of various clustering algorithms started to be considered in the context of unsupervised clustering. Thus, a variety of consensus functions were proposed to solve the ensemble clustering problem. More recently, three surveys were published [63, 100, 153] trying to summarize the field. The general idea of ensemble clustering is very simple. Consider the schematics presented in Figure 3.1. It is divided in two main processes: a) Generation step takes the original dataset and outputs an ensemble of partitions; and b) Consensus step takes the ensemble as input and outputs the final consensual partition.

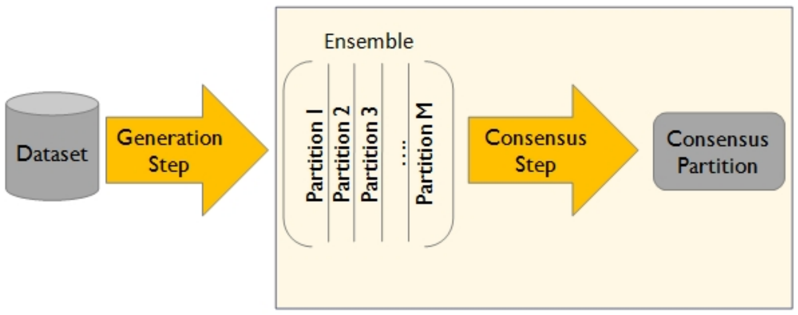


Figure 3.1. General ensemble clustering model

Later in this chapter, a more detailed version of the ensemble clustering framework is presented differentiating the two main types of consensus functions and addressing the consensus partition evaluation step.

Many authors present different reasons to use ensemble clustering techniques. It is though well accepted to take as granted that the consensual opinion of a group is more reliable than the opinion of a single individual. In this sense, ensemble clustering methods are suitable to use in situations where:

- The distribution of the dataset is unknown;
- To smooth the clustering result in cases which a suitable clustering algorithm cannot be identified;
- To improve the final clustering result, by gathering information among different partitions;

In the end, the real affirmation that can be made about ensemble clustering is that the consensus result takes into account information about all partitions in the ensemble.

Some works [57, 144] tried to define a set of properties that endorses the use of ensemble clustering methods. However, there is no agreement among them, since this is still an unanswered question. The difficulty encountered in defining the set of properties ensemble clustering methods must complain is due to the fact that most of the proposed properties are very hard to be proved. Four properties are selected given its relevance. It refers to the authors cited above for further discussion about the ensemble clustering properties.

- **Robustness** - the ensemble clustering method should present better overall performance than any of the individual clustering algorithms used to generate the partitions in the ensemble;
- **Consistency** - the consensual result must be somehow very similar to all the combined single partitions in the ensemble;
- **Novelty** - the ensemble clustering methods must be able to reach results unattainable by any traditional clustering algorithm;
- **Stability** - the consensual results must present lower sensibility to noise and outliers.

Regarding **robustness**, it is possible that in an ensemble only a subset of the partitions present good accuracy. Since the consensual result is an agreement of all partitions as stated by the **consistency** property, the robustness property in this case will most likely be violated.

This chapter reviews systematically the field of ensemble clustering. Section 1 presents a detailed model for the ensemble clustering framework. Section 2 presents a taxonomy based on the proposed classification made by Vega-Pons *et al.* [154] differentiating the various consensus functions into two different groups, namely co-occurrence and median based methods. Section 3 presents methods for consensus partition evaluation. It also addresses how to measure variability into an ensemble. Finally, a new method is introduced to visualize the variability within an ensemble based on multidimensional scaling. This section, more specifically the part regarding the visualization of the ensemble variability, presents a new contribution to the field of clustering ensemble. Consensus partition evaluation is the topic of Section 4. Section 5 presents an useful software developed to ease the process of clustering ensemble.

3.1 Detailed Ensemble Framework

Similarly to the detailed clustering frameworks proposed by Xu *et al.* [170] and Jain *et al.* [77], it is possible to devise a detailed framework for ensemble clustering. Most of the proposed ensemble clustering methods focus solely in the consensus function that will ultimately produce the consensus partition. However, some methods require that a very specific generation scheme to be followed such as the one proposed by Topchy *et al.* [145]. Another interesting remark refers to the impact the ensemble generation step has in the final consensus result. Consequently, this thesis prefer to address the problem of ensemble clustering not only by its consensus functions, but also covering the generation step and subsequently assessment of the quality of the consensus partition. The result is the detailed framework presented in Figure 3.2.

The framework receives as input a given dataset X to be clustered. No assumptions are required about the dataset distribution. Additionally to the dataset some ensemble clustering methods could require the number K of target clusters to be specified. However, there is a number of methods available not imposing such requirement.

The generation step is responsible for creating M partitions using the provided dataset as input. Different clustering algorithms, initialization parameters, or views

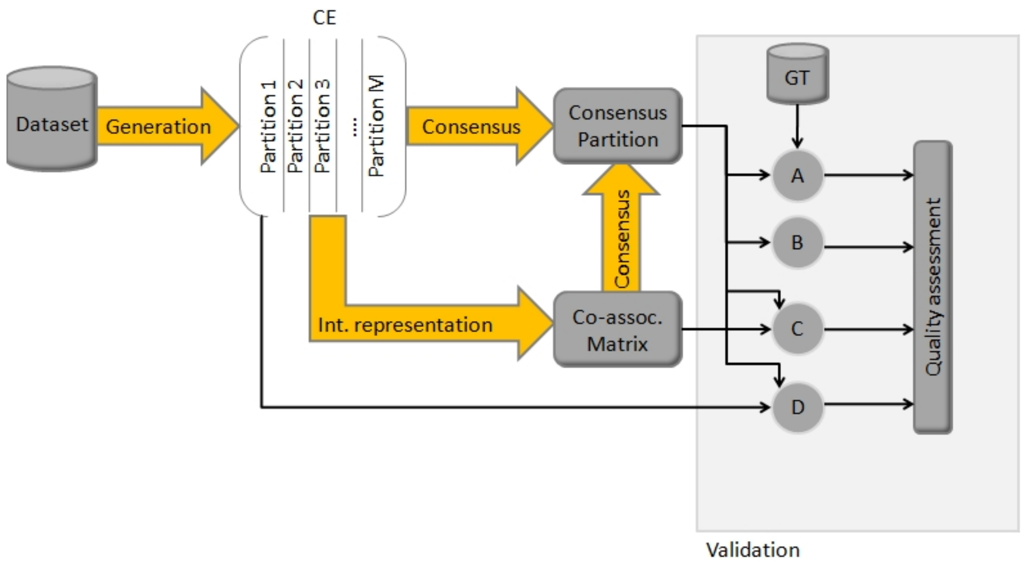


Figure 3.2. Detailed ensemble clustering framework

of the data are used in order to create an ensemble of partitions. The methods regarding the ensemble generation are presented in details in the Section 2.3.

Once the clustering ensemble is available, the consensus step takes place. Its objective is to combine all partitions in the clustering ensemble into a final consensual result also called consensus partition (CP). A detailed review of existing consensus functions is presented in Section 3.3. Alternatively, some methods require an intermediary representation of the clustering ensemble prior to the execution of the consensus function (CF). Those methods are regarded as voting methods, such as the one proposed in [57]. This particular method relies on the computation of a co-association matrix as an intermediary step. A detailed review of the voting systems can be found in in Section 3.3.

The final step regarding the ensemble clustering framework refers to the evaluation of the consensus partition. A myriad of methods exists to perform such evaluation. There are also other evaluation methods specifically designed to take advantage of the information provided by the ensemble. Section 3.4 reviews those evaluation methods.

The process of ensemble clustering is easy enough to be quickly understood. However, a number of considerations need to be made about the two main steps, namely, the generation and combination steps. Nonetheless, they are further detailed in this chapter.

3.2 Techniques for Ensemble Generation

The initial and still critical step in any ensemble clustering method refers to the ensemble generation. As the name suggests, it is the process of clustering a dataset M times to compose the ensemble of partitions. This step is critical since the result reached by any consensus function will be conditional to the information available in the ensemble.

Most consensus functions do not require any specific type of ensemble to be provided. However, there are few methods such as voting K-Means [58] that do require a well established generation mechanism. There are still methods as in [145] specifically designed to work over weak partitions. Generally, any such ensemble should suffice to the existing consensus functions.

Figure 3.3 depicts the existing ensemble generation methods, as well as how they are connected. It is also possible to combine any of the five generation ways in any desired order to create the target ensemble.

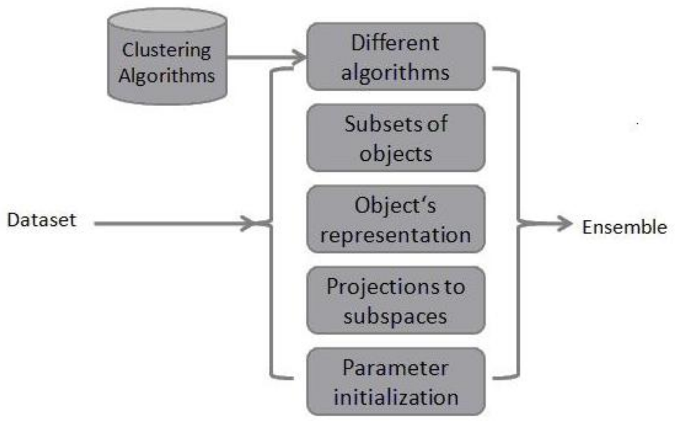


Figure 3.3. Schematics for the generation step

Ensemble generation using **different algorithms** are the first possibility. The process requires the original dataset to be provided as well as a pool of clustering algorithms. One or more partitions in the ensemble are generated using each algorithm composing this way the ensemble. The second option is to use **subsets of objects**. Instead of using all available patterns, only subsets of them are used to generate each partition. **Object's representation** is the third option. In this case, the form of each parameter is changed during the generation of each partition. A possible way to do so is to collect different information in creating each object. **Projection to subspaces** is the fourth generation method in which patterns are represented using different subsets of the available attributes. The final way to create ensembles refers to **parameter initialization**. Many clustering methods require parameter to be set in order to produce a clustering result. By varying such parameter, different partitions can be generated.

Still, the question remains: which generation method to use? The answer to that question cannot be easily obtained. In cases which additional information about the dataset is available, the generation method most likely to produce a good result is the one which produce reliable individual partitions for the specific dataset. However, when no additional information is available, the ensemble with highest probability of culminates in a good source for the consensus functions are probably the ones presenting the highest variability among partitions. However, the variability cannot be taken lightly. Each partition should also be validated using CVIs in order to assess its accuracy.

The experiments presented in this thesis utilizing clustering ensembles rely on various generation strategies as described in Chapter 2. However, in order to ensure no such two partitions very similar to appear in the ensemble, a series of tests are performed. Firstly, any new candidate partition is checked against all others already in the ensemble in order to ensure it is not much similar. This is done by means of computing the distance between the current partition and all others. A parameter Γ is specified and any partition presenting similarity lower than Γ will be discarded. This is necessary due the random nature of the generative methods described earlier. The ensemble variability visualization strategy described later in this chapter also provides an additional way to ensure the variability in any generated ensemble. After an ensemble is generated, a plotting of its variability using the MDS is performed and visually inspected. Any ensemble presenting concentration of partitions into a single region is refused.

3.2.1 Measuring Ensemble Variability

There is a direct consequence derived from the novelty property presented earlier. Consider the situation in which all partitions in the ensemble are exactly the same, which possibly only a disagreement in labels assigned to each cluster. In such case, there would be no variability in the ensemble. Therefore, it is expected by any consensus clustering algorithm to produce exactly the same partition with possibly only different labels assigned.

In the case described above, there would be no need of applying a consensus clustering method, since all results are equal. Therefore, it is reasonable to conclude that consensus methods only apply in cases which there are disagreements, or variability in the partitions composing the ensemble. In cases presenting low variability, it should be considered if the additional computational cost imposed by ensemble clustering methods pay off the potential improvement in accuracy or the smoothing factor provided by ensemble clustering methods.

Given these observations, it can be concluded that a reliable way to measure ensemble variability would play a fundamental role in the general ensemble clustering framework.

A possible way to measure ensemble variability is to compute the C_{var} index proposed in Equation (3.1) over a given ensemble $\mathbb{P}$. It is based on the SoD formulation reviewed in Chapter 2.

$$C_{var}(\mathbb{P}) = \frac{\sum_{i=1}^{M} \sum_{j=i+1}^{M} d(P_i, P_j)}{M \times (M-1)/2} \tag{3.1}$$

Provided a dissimilarity measure $d(\cdot, \cdot)$ returning values in the range $[0, 1]$, C_{var} will also provide an index in the range $[0, 1]$ indicating the degree of variability of the ensemble.

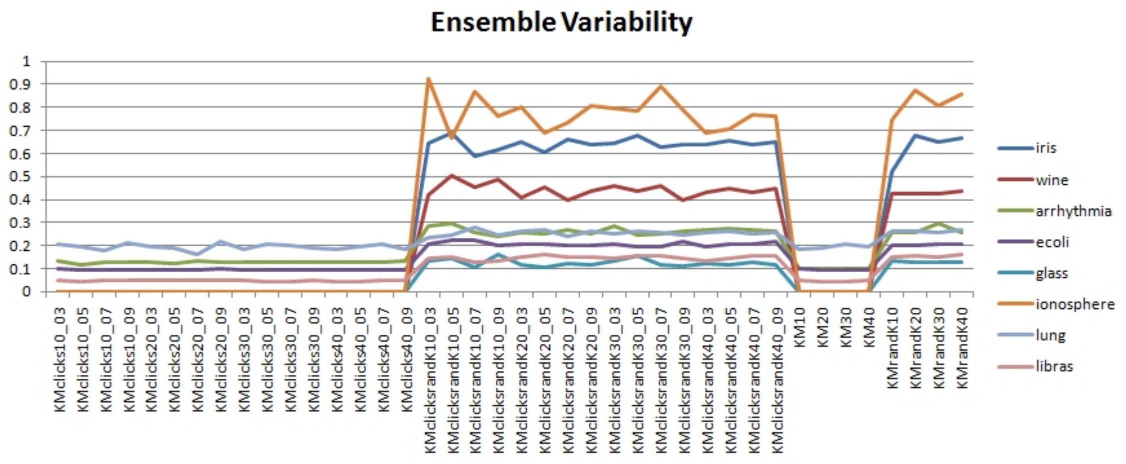


Figure 3.4. Evaluation of ensemble variability based on C_{var} index

Figure 3.4 shows the computation of C_{var} for eight datasets extracted from the UCI-Irvine database. The behavior observed on these examples is also observed in the remainder of selected datasets listed on Chapter 2. It is clear, that simple subsets of attributes (KMclicksXX-YY) as well as the reliance on random initialization (KMXX) ensembles present low variability. The low variability for the case of KMclicksXX-YY can be due to the fact the datasets in questions are composed by just a few attributes limiting the possible subsets chosen in order to create different partitions. The low variability for the case of KMXX is due to the fact K-Means induces almost always the same cluster, not mattering the random initialization of the cluster centers.

In most cases, the highest ensemble variability is found on ensembles generated using different clustering algorithms. The subsets of attributes with a random number of target clusters, such as simply random number of target clusters present also good variability.

However, such index is incapable on indicating if the subsets of partitions in the ensemble present low variability or even if they are exactly the same. It is a situation that can usually occur in using automatic ensemble generation processes. To identify such cases, it is necessary to inspect the similarity/dissimilarity between each pair of patterns individually, a process that can be extremely costly if not impossible, depending the size of the ensembles. To deal with the cases of low variability in only a subset of partitions, a visual representation of variability seems to be more suitable.

3.2.2 Visualizing Ensemble Variability

The ensemble variability plays a fundamental role in the successful generation of a reliable consensus partition. It is possible to collect evidence of such variability using a comparison between the partitions of an ensemble. However, a form of visualizing such variability can provide a simple way to inspect such variability, especially in cases which subsets of partitions present low variability when they are considered in subgroups.

Any successful visualization method should be able to encode into a single plotting the differences between all partitions in the ensemble. The major problem regarding the visualization of differences between partitions is the high dimensionality of the partitions themselves. Any two- or three-dimensional plotting of such similarities would necessary require a kind of dimensionality reduction. A plausible way to achieve dissimilarity plotting such as the ones described above can be achieved by some form of multidimensional scaling.

Multidimensional scaling (MDS) [21, 146] seems to fit perfectly to handle this task. MDS is a data analysis technique that displays the structure of distance-like data as a geometrical picture. Furthermore, it allows the embedding of a seemly higher data-space into a lower e.g. two- or three-dimensional space. The embedding reduces the problem of variability representation/inspection to a level manageable for human evaluators. MDS has its origins in psychometrics. It was proposed to help understand people's judgments of the similarity of members of a set of objects. Torgerson [146] proposed the first MDS method and coined the term.

The goal of MDS is to find an embedding of the given objects into a new lower multidimensional space in such a way that distances are preserved. It requires as input a set of objects to be embedded and a distance function suitable between the objects. The method follows by computing a matrix of size $M \times M$ (M been the number of objects in $\mathbb{P}$) containing the distances $\Delta_{i,j}$ between any two given objects. Given Δ, the objective is to find M vectors $x_1, \cdots, x_M \in \mathbb{R}^D$ such that:

$$\|x_i - x_j\| \approx \Delta_{i,j} \ \forall i, j \in \mathbb{P} \tag{3.2}$$

Usually, MDS is formulated as an optimization problem. A well known cost function is given in Equation 3.3.

$$\min_{x_1,\cdots,x_M} \sum_{i<j} (\|x_i - x_j\| - \Delta_{i,j})^2 \tag{3.3}$$

The nomenclature commonly applied in clustering ensemble is purposely used here in order to ease the application of MDS for the visualization of ensemble variability.

An example of matrix Δ is presented in Table 3.1. It is computed using the VI distance metric presented in Chapter 2. Note that in this case the classical MDS is used. Therefore, it is indicated to use a distance metric. The ensemble $\mathbb{P}$ used is comprised of 10 partitions generated over the UCI-Irvine iris dataset by means of random selection of subsets of attributes. The matrix Δ has a 10×10 size. Each column and row indexing one of the partitions in $\mathbb{P}$.

Once the matrix Δ is available, MDS can be computed to produce a dimension reduction of the data. There is a number of efficient implementations of MDS methods. In special, the Statistics Toolbox for Malab® provided functions for both classical and non-classical multidimensional scaling. The plot presented in Figure 3.5 is generated using classical MDS (the Matlab® function used to generate such plot is called "cmdscale"). It requires as input the matrix of dissimilarities Δ and the target number of dimensions to which the dimensionality reduction must be done. As output, a set of points in the new dimension is generated. Afterwards, a standard plotting function is used.

Points close to each other represent similar partitions and points farther away,

Table 3.1. Example of matrix Δ computed over an UCI-Irvine iris ensemble

0.00	0.65	0.16	2.43	0.41	0.27	1.91	0.82	1.12	0.71
0.65	0.00	0.57	2.49	0.83	0.67	1.85	0.71	0.93	0.56
0.16	0.57	0.00	2.42	0.42	0.16	1.88	0.72	1.08	0.59
2.43	2.49	2.42	0.00	2.23	2.33	2.70	2.42	2.44	2.33
0.41	0.83	0.42	2.23	0.00	0.26	1.93	0.94	1.24	0.86
0.27	0.67	0.16	2.33	0.26	0.00	1.88	0.78	1.13	0.65
1.91	1.85	1.88	2.70	1.93	1.88	0.00	1.67	1.62	1.73
0.82	0.71	0.72	2.42	0.94	0.78	1.67	0.00	0.75	0.26
1.12	0.93	1.08	2.44	1.24	1.13	1.62	0.75	0.00	0.71
0.71	0.56	0.59	2.33	0.86	0.65	1.73	0.26	0.71	0.00

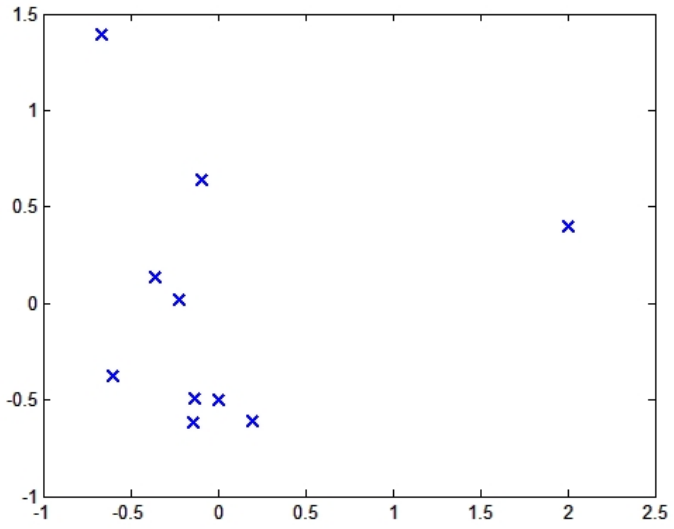


Figure 3.5. Plot of the UCI-Irvine iris ensemble using classical multidimensional scale

dissimilar partitions. The scale in both axis also plays an important role and must be analyzed in relation to the distance function used to generate the matrix Δ. In this particular case, it is possible to identify a concentration of partitions in the lower-left corner and two considerably dissimilar partitions farther away. Ideally, it is preferred that the partitions distribute themselves around a central point, fact that would ultimately provide the most information in order to a consensus partition to produce a good result.

Eventually, it could also be useful to be able to represent the ground-truth (when available) and any number of consensus partitions. Fortunately, this can be easily achieved by accounting such partitions of interest during the computation of Δ.

Figure 3.6 shows the ensemble generate for the UCI-Irvine iris dataset accounting

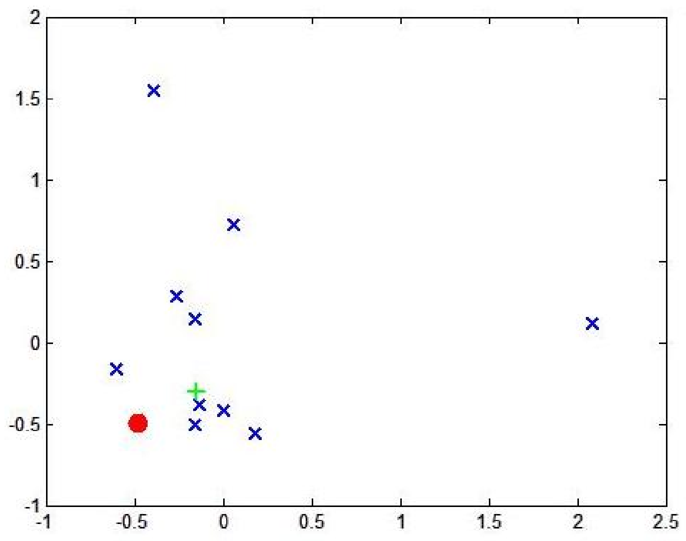


Figure 3.6. Plot of the UCI-Iris ensemble augmented by the ground-truth and consensus partition

to the available ground-truth as well as to a consensus partition. The partitions of the ensemble are represented using blue ”x” marks, the ground-truth by a red circle and the consensus partition produced using the ensemble as input by a green ”+” mark. It is possible to see that many of the partitions in the ensemble are located closer to the ground-truth. However, some partitions that can be regarded as outliers are located farther away. The consensus partition is located in the vicinities of the ground-truth, separated by approximated the same distance than some of the partitions. However, its location in the graph is shifted. This is due to the consensus function used to produce this result not only picked the best solution among the ones available in the ensemble. It also extrapolates information available in the other partitions. The final result produced is, therefore, more similar to the ground-truth.

This is an interesting fact, since it is allowed graphically to inspect how the consensus function is able to infer a novel and additionally more accurate result.

3.2.3 Impact of Different Ensemble Generation Techniques

As described before, the technique chosen for ensemble generation plays a fundamental role in the ensemble clustering framework. As stated in Chapter 2, a number of different generation schemes are proposed. Ten ensembles are created for each ensemble generation scheme and the average results are presented. This is done in order to minimize the effects of the random choices made during the ensemble's generation.

Table 3.2 presents the compared results for six ensemble clustering methods described in the next section. The values presented are given in error ratio representing the percentage of patterns wrongly classified in comparison to the known ground-truth. Those methods are selected since they represent the classes of consensus functions proposed by the taxonomy. Another reason for this choice is the fact that those specific consensus functions are commonly used throughout the literature to compare the results obtained by new consensus functions proposed.

The first remark regarding the results is the fact that the increase of the number of partitions in the ensemble does not necessarily imply an improvement in the result obtained by the consensus functions. For instance, KM_clicks20_05 and KM_clicks40_05 lead to the same result, despite the fact the second generation scheme has two times more partitions.

Since the generation schemes presented rely into random choices, there is no guarantee an ensemble can be reproduced. For instance, the subsets of attributes require that only a portion given in percentage of the attributes to be used by the clustering algorithm. Similarly, for random number of target clusters, a number is selected at random within a given range. It does not come as a surprise that the evaluation presented does not reflect the best achieved result but the average of ten ensembles generated using the same parameters. As an example, consider the case of UCI-Irvine balance dataset presented in Table 3.2. There is in fact a case for KM_clicks40_5 in which BoK reaches 21.17% but the average value presented is only 39.68%.

It is also interesting that very simple methods such as BoK and $BOEM$ can sometimes achieve results as good or even better when compared to more sophisticated ones such as graph based or evidence accumulation methods.

The same general remarks presented for the UCI-Irvine balance dataset are also observed in the other datasets evaluated in this thesis.

3.3 Taxonomy of Consensus Functions

It is very difficult to create a precise taxonomy of the existing consensus functions. The major problem resides in the fact that a multitude of methods can be potentially used to create consensus functions. However, Vega-Pons *et al.* [154] made a fairly good job in summarizing the existing methods. This work bases the review of the

Table 3.2. Consensus results for different ensemble generation techniques over the same dataset (UCI-balance)

Ensemble	*BoK*	*BOEM*	*EAC_SL*	*EAC_AL*	*HGPA*	*CSPA*
KM_clicks10_03	48.96	48.96	48.96	48.96	54.72	52.96
KM_clicks10_05	58.08	58.08	58.08	58.08	60.32	64.48
KM_clicks10_07	50.08	50.08	50.08	50.08	61.12	60.48
KM_clicks10_09	53.44	53.44	53.44	53.44	51.04	53.12
KM_clicks20_03	48.8	48.8	48.8	48.8	59.36	54.24
KM_clicks20_05	39.68	39.68	39.68	39.68	58.88	50.24
KM_clicks20_07	49.6	49.6	49.6	49.6	59.84	60.32
KM_clicks20_09	53.44	53.44	53.44	53.44	51.36	53.12
KM_clicks30_03	48.96	48.96	48.96	48.96	53.6	52.96
KM_clicks30_05	57.92	57.92	57.92	57.92	60.48	65.92
KM_clicks30_07	49.6	49.6	49.6	49.6	59.52	60.32
KM_clicks30_09	53.44	53.44	53.44	53.44	51.2	53.12
KM_clicks40_03	48.8	48.8	48.8	48.8	58.88	54.24
KM_clicks40_05	39.68	39.68	39.68	39.68	52.8	50.24
KM_clicks40_07	50.08	50.08	50.08	50.08	59.68	62.08
KM_clicks40_09	53.44	53.44	53.44	53.44	50.4	53.12
KM_clicksrandomK10_03	37.76	37.76	48.48	48.48	58.88	51.52
KM_clicksrandomK10_05	57.92	57.92	57.92	57.92	64.16	61.12
KM_clicksrandomK10_07	65.92	65.92	54.56	52.16	57.6	58.88
KM_clicksrandomK10_09	65.92	65.92	54.56	52.16	56.32	58.88
KM_clicksrandomK20_03	63.52	65.28	53.6	53.76	60.32	59.84
KM_clicksrandomK20_05	49.12	49.12	49.12	49.12	61.6	51.68
KM_clicksrandomK20_07	39.68	39.68	33.28	39.68	57.28	46.08
KM_clicksrandomK20_09	45.12	45.12	48.48	52.0	60.0	54.08
KM_clicksrandomK30_03	53.44	53.92	56.16	54.56	60.16	52.96
KM_clicksrandomK30_05	38.24	38.24	48.96	48.96	62.56	52.64
KM_clicksrandomK30_07	58.88	58.88	54.56	58.08	60	65.6
KM_clicksrandomK30_09	45.6	45.6	50.4	55.04	59.68	49.92
KM_clicksrandomK40_03	53.6	53.6	53.6	56.16	58.56	54.56
KM_clicksrandomK40_05	38.24	38.24	48.96	45.12	64	52.64
KM_clicksrandomK40_07	40.32	40.32	33.76	40.16	60.32	45.12
KM_clicksrandomK40_09	49.6	49.6	48.32	49.28	60.64	58.56
KM_randomK10	53.44	53.44	53.44	53.44	50.72	53.12
KM_randomK20	63.52	64.8	46.4	54.56	57.28	52.16
KM_randomK30	53.6	53.6	53.6	55.36	58.4	53.6
KM_randomK40	53.6	53.6	54.72	53.76	60.16	58.24

existing consensus functions on such work.

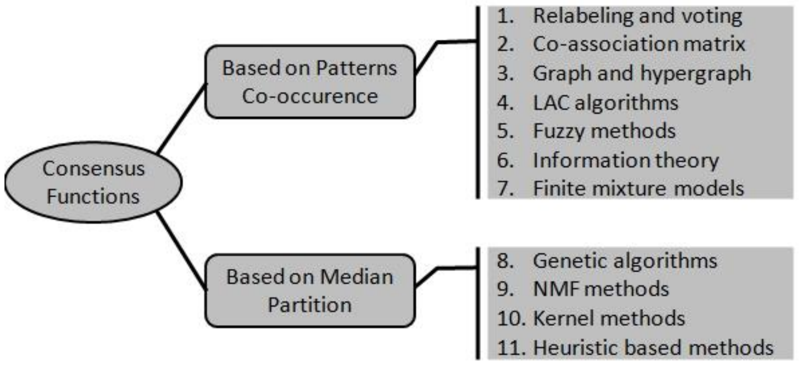


Figure 3.7. Simplified taxonomy of ensemble clustering methods

As Figure 3.7 shows, there are two main strains of consensus functions, namely methods based on the patterns co-occurrence and based on the median partition formulation.

3.3.1 Median Partition Based Methods

One of the most popular formulations for the consensus clustering problem is based on the median partition (MP) problem [50]. The median partition is defined as the one that minimizes the sum of distances between it and all the partitions in the ensemble. It can be formally stated as follows:

Given M partitions $P_1, \cdots, P_M$ and a distance $d(\cdot, \cdot)$, which is a symmetric, find P^* such that:

$$P^* = \arg\min_{P_i} \sum_{i=1}^{M} d(P_i, P) \tag{3.4}$$

This problem is known to be $\mathcal{NP}$-complete [50], for many reasonable distance functions, directing the research to the development of heuristics to approximate it.

(A) Genetic Algorithms: As the name suggests, the GA formulation is used to, throughout the evolution of consecutive generations, infer the consensual partition. Most methods, in this class, uses the information available in the ensemble to create initial populations of potential consensual partitions. After each population is created, the "chromosomes" are evaluated by a fitness function and the fittest ones are used to feed the mutation, crossover, and any other genetic operators.

A plausible fitness function is presented in Equation (3.5). It evaluates if chromosome $\tilde{P}$ (candidate consensus partition) has a SoD value in relation to the partitions in $\mathbb{P}$, smaller or equal than an acceptance threshold parameter Γ. If this is the case, this particular chromosome will be selected to reproduction. The population size N_{pop} is a free parameter.

$$SoD(\mathbb{P}) = \sum_{i=1}^{N_{pop}} d(P_i, P) \leq \Gamma \tag{3.5}$$

Regarding the generation of the initial population, a number of different generation methods [106, 173] can be applied. A common method refers to perform mutations over the initial partitions in $\mathbb{P}$, or simply consider the clustering ensemble as initial population.

(B) Nonnegative Matrix Factorization Based Methods: NMF-methods refer to the method of factorizing a nonnegative matrix R into two matrix factors, i.e. $R \approx AB$. A and B must also be nonnegative. An example of such method can be found in [101].

(C) Kernel methods: Vega-Pons *et al.* [154] introduced a Kernel based method throughout the Weighted Partition Consensus via Kernels (WPCK) algorithm. In this method, the consensus partition is defined as follows:

$$SoD(\mathbb{P}) = \arg\max_{P \in \mathbb{P}} \sum_{i=1}^{M} \omega_i \cdot \hat{k}(P, P_i) \tag{3.6}$$

where ω_i is a weight associated to partition P_i and $\hat{k}$ is a similarity measure between partitions, which is a kernel function [134].

The weight values ω_i are usually computed in a step previous the combination, which the relevance of each partition is estimated by the application of several internal validity indexes. However, in this paper, the authors do not consider the weights values because their computation needs the use of the original data. Then, the values are set to $\omega_i = 1; \forall\, i = 1, \cdots, M$.

The kernel property of $\hat{k}$ allows mapping this problem into a Hilbert space $\mathcal{H}$, that an exact solution can be easily obtained. Given the solution in $\mathcal{H}$ the pre-image problem could be solved, i.e., finding the partition in $\mathbb{P}$ which corresponds with the solution in $\mathcal{H}$. This is usually a hard optimization problem that could not have an exact solution. The simulated annealing meta-heuristic was used to obtain an approximated solution avoiding the convergence to local minima. In this algorithm, the specification of the number of clusters in the final partition is not necessary. Partitions with different number of clusters are generated and analyzed in the simulated annealing process. After a stopping criterion is reached, the best partition obtained by the process is returned as the final result. However, this algorithm can be modified to work with a specified number of clusters K in the final partition. In the case of a fixed K, from the best partition in the cluster ensemble, we generate a partition with K clusters making the minimum number of movements of objects from one cluster to another. After that, the simulated annealing is applied but only considering as new states in the process, partitions with K clusters.

(D) Heuristic Based Methods: Among the relevant works in proposing heuristics to solve ensemble clustering via the median partition formulation, Golder and Filkov [67] present a collection of six heuristics. The simplest heuristic proposed is essentially a selection process, known as BoK. The idea behind Best of K is to select the best or, most representative partition among all partitions in the ensemble. This is achieved by selecting iteratively each partition in $\mathbb{P}$ and computing the sum of distances (SoD) between the selected partition and the remaining ones in the ensemble. The partition in the ensemble with smaller SoD value is selected as consensus partition. As it can be seen by inspecting Equation (3.7), it is essentially the set median.

$$P^* = BoK(\mathbb{P}) = \arg\min_{P \in \mathbb{P}} \sum_{i=1}^{M} d(P_i, \bar{P}) \tag{3.7}$$

A second heuristic proposed is known as $BOEM$ (The Best One Element Moves) It starts with an initial consensus clustering partition. Any partition $P \in \mathbb{P}$ can be selected as initial result. The algorithm follows by interactively testing each possible label for each pattern, retaining the label that decreases the SoD.

(E) Clustering Based on Semi Definite Programming: SDP [136] is motivated by the observation, that pairwise similarity values between patterns as used in [140] do not provide sufficient information for ensemble clustering algorithms. Therefore, the authors propose to encode the solutions obtained by individual clustering results by a multidimensional string. In the first step for every data element a so called A-string is computed, which encodes the information from the individual clustering results. The ensemble clustering problem reduces to a form of string clustering problem which the objective is to cluster similar strings to the same cluster. For this reason the authors first formulate a non-linear objective function, that is transformed into a $[0-1]$ semi definite program (SDP) using a convexification technique. Thus, this program is relaxed to a polynomial time solvable SDP.

Finally, random walker method [1] creates a graph representation of the data set, specifies weights for each edge, defines start seed vertices and then, it applies na heuristic based on random walker to infer the consensual partition. This method will be further explained in details later on Chapter 8.

3.3.2 Patterns Co-occurrence Based Methods

In [57], the authors explored the idea of evidence accumulation by combining the partitions generated by of M tries of K-Means into a co-association matrix. This matrix is later used as a new similarity measure for a standard agglomerative hierarchical clustering algorithm. The method can be divided in two steps.

(A) Relabeling and Voting: Methods based on relabeling and voting rely on the idea of firstly solve the label correspondence problem. Once the label correspondence of all partitions in the ensemble is solved a voting process decides the consensus partition. Various clustering ensemble methods [51, 149, 163] try to solve the label correspondence problem using different heuristics, such as the cumulative voting or bipartite graph matching. Another valid approach is to use the Hungarian algorithm [55].

Ayad and Kamel [7] proposes three new cumulative voting methods. The problem is formulated as finding a compressed summary of the estimated distribution that preserves the maximum relevance. It starts selecting a reference clustering based on the maximization of the intra-cluster Shannon entropy. A mapping is computed between the elected reference partition and the remainder partitions. Due to a probabilistic mapping, the problem of label assignment is elegantly addressed. Then, a re-mapping is computed generating a new divergence matrix and the consensus clustering is obtained via an adapted version of the AIB algorithm. Finally, in [162] it is presented a combination approach based on a random walker algorithm to perform fusion of multiple image segmentations.

(B) Evidence Accumulation, Co-occurrence Matrix: The underlying assumption is based on the fact that objects belonging to the same "natural" cluster are very likely to be collocated in the same cluster among different partitions. A co-occurrence matrix CM of size $N \times N$ with values ranging from 0 (meaning no association) to 1 (maximum association) is computed using Equation (3.8).

$$CM(i,j) = \frac{m_{i,j}}{M} \tag{3.8}$$

where $m_{i,j}$ refers to how many times the pair (i,j) of objects occurs in the same cluster among the M partitions.

After computing the co-association matrix the general clusters are most-likely to be found orbiting the principal diagonal. The job of consensus function based on this method is to solve the minor disagreements occurring in other regions of the co-association matrix. Figure 3.8 presents the plotting of three co-association matrices ((A) - Wine, (B) Iris, and (C) Optic datasets).

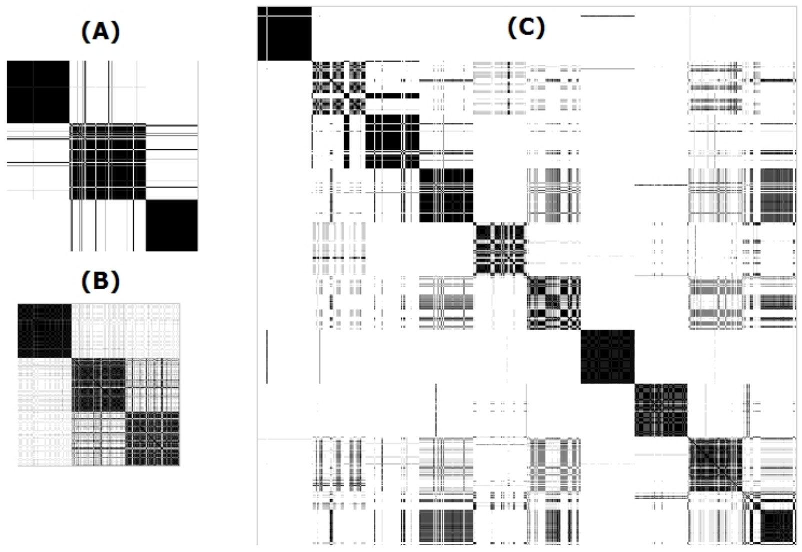


Figure 3.8. Plotting of three co-association matrices

For the combination step, this method uses yet another clustering algorithm, more specifically, a hierarchical clustering algorithm such as single-link or average-link. The co-association matrix CM is regarded as a new data space and used as input for the hierarchical algorithm. The result produced is the consensus partition.

(C) Graph and Hypergraph: Strehl *et al.* [140] presents one of the first works in the area of unsupervised ensemble clustering. In this work, three graph based on heuristics are proposed, namely CSPA, HGPA and MCLA. Those heuristics represent the clustering ensemble as a hypergraph in which each partition is encoded as a hyperedge.

Cluster-based Similarity Partitioning Algorithm (CSPA). In this method, an $N \times N$ similarity matrix is defined from the hypergraph. This can be viewed as the adjacency matrix of a fully connected graph, which the nodes are the elements of the set X (original dataset) and an edge between two objects has an associated weight equal to the number of times the objects are in the same cluster. Then, the graph partitioning algorithm METIS [85] is used to obtain the consensus partition.

HyperGraphs Partitioning Algorithm (HGPA). This method partitions the hypergraph directly by eliminating the minimal number of hyperedges. It is considered that all hyperedges have the same weight, and it is searched by cutting the minimum possible number of hyperedges that partitions the hypergraph K connected components of approximately the same dimension. For the implementation of the method, the hypergraphs partitioning package HMETIS [86] is used.

Meta-CLustering Algorithm (MCLA). In this method, the similarity between two clusters is defined first in terms of the amount of objects grouped in both, using the Jaccard index. Then, a similarity matrix between clusters is formed which represents the adjacency matrix of the graph. It is built by considering the clusters as nodes and assigning a weight to the edge between two nodes, whereas the weight represents the similarity between the clusters. This graph is partitioned using the METIS [85] algorithm and the obtained clusters are called meta-clusters.

(D) Locally Adaptive Clustering Algorithms: As the name suggests, this class of consensus function works over partitions produced using locally adaptive clustering algorithms proposed by Domeniconi *et al.* [42]. The same main author proposes three consensus functions [41], Weighty Similarity Partition Algorithm (WSPA), Weighty Bipartite Partition Algorithm (WBPA) and Weighted Subspace Bipartite Partitioning Algorithm (WSBPA).

(E) Fuzzy Methods: This class of ensemble clustering methods differently from the others reviewed in this chapter, works with ensembles composed by fuzzy partitions. It is obviously possible to harden the fuzzy partitions and therefore, it applies any other ensemble clustering method. However, valuable fuzzy information would be lost. To deal with fuzzy partitions directly, Punera and Ghosh [127] propose fuzzy versions of the methods presented in [140].

(F) Information Theory: Topchy *et al.* [145] introduced the idea of information theory via the algorithm called Quadratic Mutual Information (QMI). In this method, the category utility function [66] U is used as a similarity measure between two partitions. In this case, the category utility function $U(P_i, P_j)$ can be interpreted as the difference between the prediction of the clusters of a partition P_i both with the knowledge of the partition P_j and without it. The better the agreement between the two partitions, higher values of the category utility function are obtained. Hence, the consensus partition could be defined by using U as a similarity measure between partitions:

$$P^* = \arg\min_{P \in \mathbb{P}} \sum_{i=1}^{M} U(P, P_i) \tag{3.9}$$

It has been proven [115] that maximization of utility function is equivalent to minimization of the square-error clustering criterion if the number of clusters K is known for the consensus partition. This way the solution of the problem (Equation (3.9)) is approached as following. First, for each object the values of new features are computed using the information in the cluster ensemble. After that, the final partition is obtained by applying the K-Means algorithm on the new data.

(G) Finite Mixture Models: The concept of using the finite mixture models was explored by Topchy *et al.* [144]. This work proposes a new consensus function which the consensus partition is estimated as a solution of the maximum likelihood estimation problem.

This section presented a short review of the most relevant consensus functions. In order to assess the quality of the results achieved by such methods, a number of evaluation schemes were already proposed and are commonly used. In the next section, such evaluation methodologies are briefly reviewed.

3.4 Consensus Partition Evaluation

The common procedure concerning the evaluation of clustering results relies in the usage of CVIs (Cluster Validity Indexes). In cases where different clustering algorithms over the same dataset are used, there is a chance considerably dissimilar results will emerge. In such case, there is the need to validate the accuracy of the results. CVI indexes perform a direct comparison between the clustering result, and a partition assumed to be the "correct answer" or ground-truth. However, in many real life applications, such ground-truth is rarely available. Therefore, an evaluation scheme considering only the data available is needed. Furthermore, with the introduction of ensemble clustering, although possible to still use the evaluation methods developed for classical clustering, extra information encoded in the ensemble of partitions is available, therefore, it seems logical to take into account such information, in order to perform a better, more accurate evaluation. This section reviews such evaluation schemes. Afterwards, Chapter 7 presents the concept of sum of pairwise distances, and further develops a new evaluation measure based on this formulation.

The related literature lists three main types of cluster validity indexes: a) external; b) relative; and c) internal indexes [23].

External Indexes: This type of index uses external information about the data to validate the partition in question. The external information is usually translated as a ground-truth. The criterion of quality for this kind of index is related to the degree of similarity to the external comparison source. Typically, any valid similarity measure suitable for partitions comparison can be used as an external CVI. All indexes/similarity measures revised in Chapter 2 are valid options to be used as external CVIs.

Relative Indexes: This type of index relies on the idea of comparing the partitions obtained using the same clustering algorithm under different conditions. The assumption of this method is that by comparing such partitions, the most accurate one can be inferred. Among the most relevant CVIs in this class are Figure of Merit [171] and Stability Index [131].

Internal Indexes: This type of index [70] does not require an external source of information, neither different partitions produced by the same algorithm under different circumstances as the previous two described types. Instead, it relies simply on the information available into the clusters. It measures characteristics such as inter and intra and inter-cluster distances, separation, connectivity, mean size of clusters, and difference between the cluster sizes.

Any cluster validity index provides only a relative measure of quality of the partition in question. For this reason any clustering analysis will be faced almost always with some degree of uncertainty. It is believed that this is one of the most important reasons to apply clustering ensemble techniques. It provides a compromise between the selection of a single partition which presents the most suitable CVI and the consensus of all partitions available.

3.4.1 Ensemble Clustering Validity Indexes

Much was done on validating partitions produced by traditional clustering algorithms [69]. However, the area still lacks of extensive considerations regarding the specific evaluation of consensus partitions. Classical cluster validity indexes can be used to assess the accuracy of partitions produced by clustering ensemble methods. However they do not consider the extra information encoded in the clustering ensemble. Recently, new cluster validity indexes accounting for the information in the clustering ensemble were introduced.

Average Cluster Consistency: ACC [44] is a criterion based on the likelihood estimative. It is in fact an adaptation of ”classical” cluster validity indexes to pairwise similarity representations. It is defined as follows:

$$sim(P^*, \mathbb{P}) = \frac{1}{N} \sum_{i=1}^{k^l} \max_{1 \leq k \leq K} \left| C_k^* \cap C_i^l \right| \left(1 - \frac{|C_k^*|}{N} \right) \tag{3.10}$$

where $k^l \geq k^*$

$$ACC(P^*, \mathbb{P}) = \frac{1}{N} \sum_{i=1}^{N} sim(P^*, P_i) \tag{3.11}$$

Average Normalized Mutual Information: ANMI [140] is an adaptation of the well known mutual information to measure the degree of agreement of an obtained consensus partition P^* against the clustering ensemble $\mathbb{P}$. This measure simply computes the average of MI between the consensus partition and all the partitions in the ensemble. Therefore, any such dissimilarity function reviewed in Chapter 2 could potentially be used in such way.

$$ANMI(P^*, \mathbb{P}) = \frac{1}{N} \sum_{i=1}^{N} \frac{MI(P^*, P^l)}{H(P^*)H(P^l)} \tag{3.12}$$

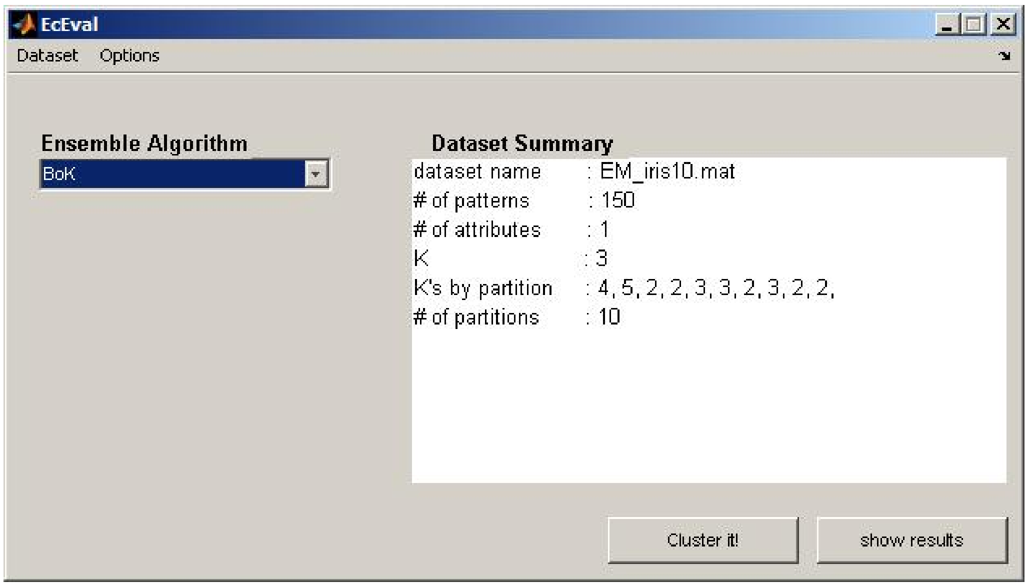
$$H(P) = p(k) \log p(l) \tag{3.13}$$

Recently, a new CVI named pairwise similarity [45] was specifically designed to work with evidence accumulation methods. This validity index is based on the likelihood of the data set given a co-association matrix computed using the partitions of an ensemble.

3.5 Ensemble Clustering Software

This section describes a simple user interface with some build-in consensus functions able to execute ensemble clustering as well to perform some minor evaluation over the obtained result. The program here introduced can be easily extended to accommodate new developed consensus functions. The software is developed using Malab® *v*.7.1 SP(13) and all the data structures used are compatible with this software.

The main interface (Figure 3.9) can be opened by typing the program's name $\ll$ *ECEval* $\gg$ in the command window. After starting, it shows a popup menu listing the consensus functions known to the program and defined in a file called $\ll$ *consensusFunctions.txt* $\gg$. This file should be located in the same directory as the program.



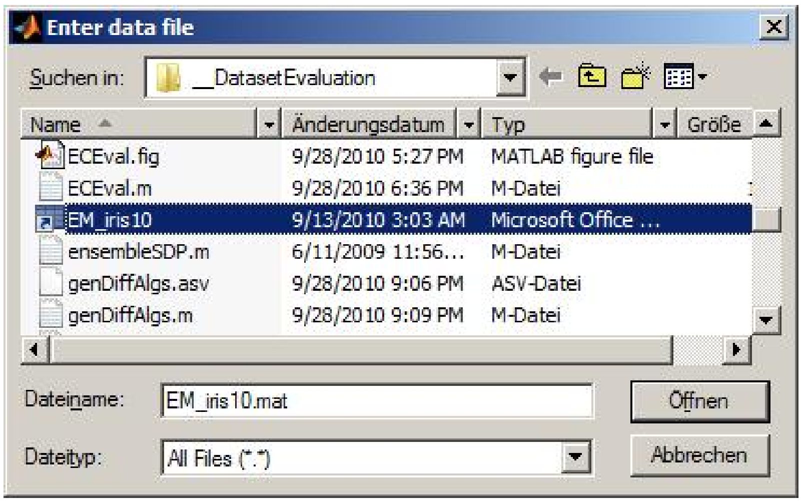
```
dataset name      : EM_iris10.mat
# of patterns     : 150
# of attributes   : 1
K                 : 3
K's by partition  : 4, 5, 2, 2, 3, 3, 2, 3, 2, 2,
# of partitions   : 10
```

Figure 3.9. Main interface of the ensemble clustering program

Additionally, since most consensus functions simply require a clustering ensemble to be provided as input, the program was built to execute the selected consensus function with this single value. However, some consensus functions such as the one based on random walker require extra parameters to be defined. In such cases, an additional control file must be created and put in the same directory of the program. The name assigned to the fine must be precisely the same specified in the $\ll$ *consensusFunctions.txt* $\gg$ file. As an example, the file $\ll$ *rw.txt* $\gg$ controls the execution of this particular consensus function. The contents of the file must be the name of the variables to be passed as parameter to the consensus function.

The next step refers to open an ensemble clustering dataset in order to the program be able to execute any known consensus function. This can be done by selecting the menu item $\ll$ *Dataset – load ensemble* $\gg$. This will bring up the interface shown in Figure 3.10. The program expects a Matlab® ".mat" to be provided as input. It must contain a variable called EM referring to the clustering ensemble ($N \times M$ size where N is the number of patterns and M is the number of partitions). Additionally, any values needed for the execution of a specific algorithm must be located here as well, with names matching the one specified in the file $\ll$ *CF_name.txt* $\gg$.

Once the program is provided with an input data file, the right box in the main



Enter data file

Suchen in: __DatasetEvaluation

Name	Änderungsdatum	Typ	Größe
ECEval.fig	9/28/2010 5:27 PM	MATLAB figure file	
ECEval.m	9/28/2010 6:36 PM	M-Datei	
EM_iris10	9/13/2010 3:03 AM	Microsoft Office ...	
ensembleSDP.m	6/11/2009 11:56...	M-Datei	
genDiffAlgs.asv	9/28/2010 9:06 PM	ASV-Datei	
genDiffAlgs.m	9/28/2010 9:09 PM	M-Datei	

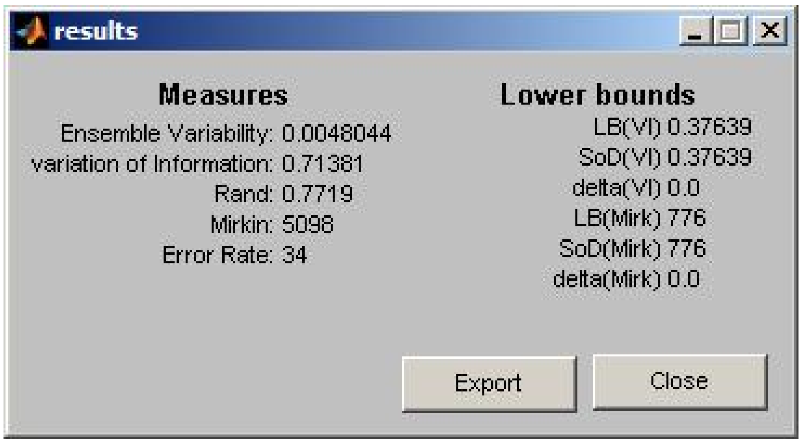
Dateiname: EM_iris10.mat

Dateityp: All Files (*.*)

Figure 3.10. Opening ensemble interface

window will display the information contained inside it. Afterwards, a consensus function can be executed. This is done by selecting the desired consensus function in the left popup menu and clicking the button ≪ *Cluster it*≫.

The final option regards the evaluation of the produced consensus partition. This can be done by clicking the button ≪ *Show results* ≫. The window shown in Figure 3.11 will be opened listing the values for various CVIs such as the lower bounds for the given ensemble.



results

Measures

Ensemble Variability: 0.0048044
variation of Information: 0.71381
Rand: 0.7719
Mirkin: 5098
Error Rate: 34

Lower bounds

LB(VI) 0.37639
SoD(VI) 0.37639
delta(VI) 0.0
LB(Mirk) 776
SoD(Mirk) 776
delta(Mirk) 0.0

Figure 3.11. Simple interface showing results for distance measures and lower bounds

This chapter reviewed the ensemble clustering method. It also presented a detailed discussion about ensemble variability and introduced a way to inspect ensemble variability intra-ensemble by means of a visualization scheme. The visualization scheme proposed based on multidimensional scaling allows the representation of each partition as a point in a 2D space, in which similar partitions are represented by points close to each other and dissimilar partitions as points far apart. As discussed in [161], the ensemble variability plays an important role in the possible improvements in accuracy achieved by an ensemble clustering method. However, few works have been done in respect to the proper assessment of such variability. Scalar indexes such as the one proposed in Section 3.2.1, although allowing the measurement of such variability in a macro scale (at ensemble level) does not allow the assessment the fact that a subset of partitions in the ensemble can present low variability. It is believed that the visualization scheme proposed poses as an interesting alternative to address such fundamental question. As a future work regarding this issue, it is possible to use the MDS formulation to identify such similar partitions and incorporate a subroutine into the generation framework to automatically regenerate partitions presenting low variability when compared with the remainder of the ensemble. This is in fact similar to what has being done in this thesis, except for the identification of similar partitions is done by visual inspection.

Another interesting topic presented by this chapter refers to the extensive evaluation of different ensemble generation strategies. The assessment of 40 different ensemble generation methods allows high confidence in the average results presented.

Finally, the ensemble clustering user interface proposed allows simple usage of the most common consensus functions available with the possibility of easy extension to work with new consensus functions still yet to be proposed. By means of this software, further benchmarks comparing the advantages and drawbacks of different consensus functions can be conceivably made. It also provides a simple coding, suitable for an introduction to the field of ensemble clustering.

Chapter 4

Random Walker Ensemble Clustering

This chapter presents the adaptation of a consensus function based on random walker to work with general clustering problems [1]. It was originally developed for combination of image segmentations [162]. This consensus function has proven to be very effective in solving the problem of combining multiple segmentations into a single consensual result. However, the work proposed in [162] tunes the algorithm to deal specifically with image datasets which is in fact a subset of the more general problem. In order to adjust the random walker consensus function, a number of issues need to be addressed. More specifically, the ensemble of partitions needs to be pre-processed to generate a graph representation. A straightforward approach such as considering a complete graph could lead to a computationally intense implementation. A new problem arises, namely what is considered to be a sufficient and/or necessary neighborhood size. It is shown experimentally that a very small neighborhood produces similar results compared to larger choices. This fact alone improves the computational time needed to produce the final consensual partition. This chapter also presents an experimental comparison between the random walker consensus function to other graph based methods in order to assess the accuracy of this approach.

The remainder of this chapter is organized as follows. Section 1 discusses the need of an adaptation of the random walker consensus function, to work with ensemble clustering problems. In order to achieve it, the ensemble of partitions needs to be pre-processed to generate its graph representation. Section 2 presents the method devised for generation of the graph representation. Afterwards, an experiment compares the results of the random walker consensus function to graph based and other well known combination clustering methods. The impact of the neighborhood's size is investigated as well as the quality of the consensus results achieved. It also shows experimentally that, a very small neighborhood produces similar results compared to larger choices. This fact has a direct impact in the processing time required by the algorithm.

4.1 Summary

To better explain the changes needed to adapt the random walker based image segmentation combination method to address general clustering problems, a short review of the original work is in order. The first step is the creation of an ensemble. During this step, different segmentation algorithms and/or different parameter settings are used to create an ensemble of segmentations.

Once the ensemble of segmentations is available, a consensus step combines them all into a final consensual segmentation. The consensus step can be divided into 3 parts: a) graph generation; b) seed region generation; and c) ensemble combination.

The original method performs a rescale of the images in order to reduce the number of pixels to be processed. Afterwards a 4-neighborhood is computed creating the graph representation needed. This graph representation is the input data expected by the next step. For more information about the graph generation in the original method, Wattuya *et al.* [162] will be referred.

The seed region generation is divided in two steps: a) candidate seed region extraction; and b) grouping of candidate seed regions. By means of these two steps automatically generated seed points, the random walker consensus function is able to automatically decide the final number of target clusters.

Seed Region Generation

A new graph $\mathcal{G}^*$ is built by preserving those edges with weight $w_{i,j} = 1$ (only p_i and p_j share the same label) and removing all other edges. This step basically retains those edges between two neighboring nodes which are most likely to belong to the same region. Then, all connected subgraphs in G* are removed and they are regard as a set of initial seeds which are further reduced in the next step.

Grouping candidate seed region

The number of candidate seed regions from the last step is typically higher than the true number of regions in an input image. Thus, a further reduction is performed by iteratively selecting the two candidate seed regions with the highest similarity value and merging them to build one single (possibly spatially disconnected) candidate seed region. Subsequently, the similarity values between a new merged region and all remaining candidate seed regions are recomputed. The similarity $s(C_i, C_j)$ between candidate seed regions C_i and C_j is defined in Equation (4.1).

$$s(C_i, C_j) = \overline{\left\{ \frac{n_{i,j}}{N} | (p_i, p_j) \in C_i \times C_j \right\}} \tag{4.1}$$

where $\bar{A}$ denotes the average of the set A.

Ensemble Combination

Given the graph $\mathcal{G}$ constructed from the initial segmentations and K seed regions, the random walker consensus function [68] is applied to compute the final segmentation. The computation of random walker probabilities can be exactly performed without the simulation of random walks. It can be achieved by solving a sparse, symmetric, positive-definite system of equations. Each unseeded pattern is then assigned a K-tuple vector, specifying the probability that a random walker starting from that pattern will first reach each of the K seed regions. A final segmentation is derived by assigning each pixel the label with largest probability.

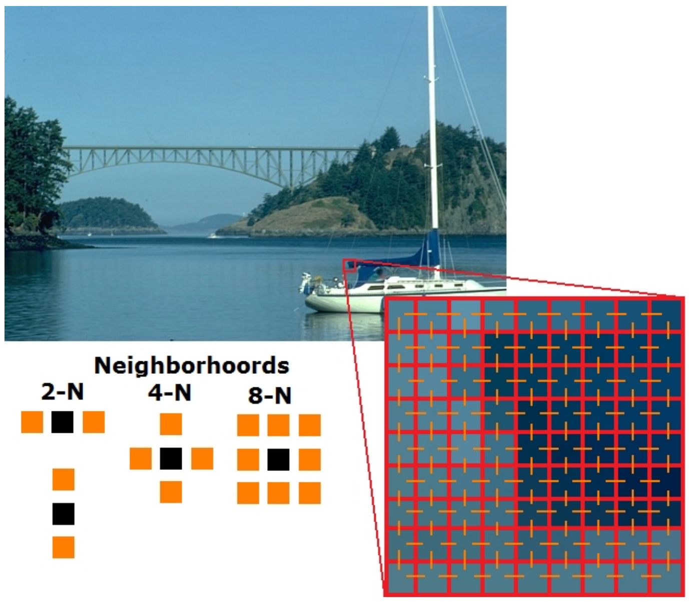


Figure 4.1. Example of well defined topology in image datasets

Random Walker based algorithms requires that a graph representation of the dataset to allow the simulation of the random walks. A simple way to create this graph representation is to use a complete graph. For this case, the distances (weights) between all possible pairs of patterns should be included. However, such representation is not efficient, especially in dealing with mid- to large-size datasets. A possible example is image datasets or general datasets with more than thousands patterns. As discussed in [54], the dimensionality problem can be minimized by combining objects (pixels) belonging to the same segmented region into a single representative. Another option [162] is to resize the image to a more manageable size. Taking advantage of the well behaved topology of images (see Figure 4.1) it is possible to define neighborhoods in a simple way. For those cases, there is no need to include in the graph representation edges linking non-neighbor regions, since they are most likely not to belong to the same structure. Figure 4.1 shows an example from the Berkeley image database which a small region is enlarged. It is clear how well the neighborhood is organized. This is a property of 2D image datasets. General clustering problems often presents patterns with a considerably higher dimensionality. Due to the high dimensionality of general data, the specification of neighborhoods is problematic.

This chapter main motivation is to propose and evaluate a feasible way to create compact neighborhood representations. Subsequently, the graph created is supplied as input to the random walker consensus function. A feasible option to the graph generation problem is to consider only a given number of closest nearest neighbors.

4.2 Graph Generation Method

High dimension datasets do not have a simple neighborhood lattice. A possible way to create its graph representation is to compute the distance between all possible pairs of points leading to a complete graph. The disadvantage of this option is twofold. Due to the fact that medium to large datasets lead to large numbers of edges ($n(n-1)/2$ edges), the use of the set of complete neighborhoods can be prejudicial to the random walker algorithm. All possible paths would have to be considered, leading to a longer processing time. Since two patterns very far apart are most likely not to be into the same cluster, it is clear that reasonable neighborhood size in which the closest neighbors to each given pattern are represented could benefit greatly the outcome of the algorithm.

Therefore, to avoid the complete graph solution, the neighborhood of each individual pattern need to be identified. Furthermore, a suitable neighborhood size, or the number of patterns to be considered as neighbors, needs to be specified.

The random walker consensus function requires as input an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$ where $\mathcal{V}$ is a list of vertices, one for each pattern existing in the dataset X, $\mathcal{E}$ is a set of edges where $e \in \mathcal{E}/e = (p, q) \wedge p, q \in \mathcal{V}$, and w is a set of weights associated to each edge in $\mathcal{E}$.

The generation of the set of vertices is straightforward: It simply defines a vertex corresponding to each pattern $p \in X$.

Edges's Computation

To generate the set of edges, firstly it needs to be decided the neighborhood size. A previous step is required in order to find the δ-nearest neighbors of a given pattern p. A distance function $d(\cdot, \cdot)$ also needs to be specified. In the experiments presented in this chapter, the Euclidean distance is used.

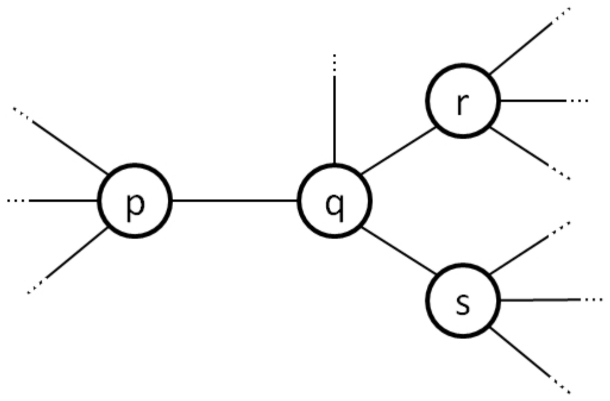


Figure 4.2. Example of neighboring patterns sharing the same edge

First, a matrix containing the pairwise distance between all possible pairs of patterns is created (naive implementation). Subsequently, the distance matrix is inspected for each pattern in order to retrieve its δ-nearest neighbors. The pairs composed by $\{p, \delta(nn_i)\}$ are inserted in the set of edges $\mathcal{E}$. Special attention needs to be paid at this step. It is possible that two patterns e.g. p and q share one or more edges. This case usually happens between two neighboring patterns. For that reason, before adding any edge to the set $\mathcal{E}$, a test is required to ensure the edges do not exist already in $\mathcal{E}$. Consider the example given in Figure 4.2. In this case, the δ-neighborhood was set to 4, meaning that for each pattern, the 4 closest patterns are selected to be added to the set of edges $\mathcal{E}$. Patterns p and q share a common edge, since q is among the δ-NN of p and vice versa. Therefore, there is no need to include the same edge twice in $\mathcal{E}$.

The random walker consensus function requires a connected graph to work properly. Therefore, the connectedness of the set of edges needs to be tested. This is not an issue in dealing with 2D image datasets. However, for the general case, there is a chance disconnected graphs will be generated. As defined in graph theory, an undirected graph $\mathcal{G}$ is said to be connected if all pairs of vertices in the graph are connected, and two vertices are said to be connected if there is a path between them.

The part of the algorithm responsible for making $\mathcal{G}$ connected is divided in three parts.

- **Test for Connectedness** - This initial test identify the need for connecting possible subgraphs. To check for connectedness is very simple. The pseudo-algorithm for this step is given by the three simple steps as follows:
  1. Start by selecting a vertex at random;
  2. Proceed from that vertex using depth-first or breadth-first search, counting the visited nodes;
  3. After traversing the entire graph, it is necessary to check if the number of counted nodes equals the total number of nodes. If it is affirmative, the graph is connected, otherwise it has two or more subgraphs.
- **Identify Subgraphs** - If the graph $\mathcal{G}$ is not connected, the list of subgraphs can be retrieved by the following pseudo-algorithm:
  1. While there are unvisited vertices, selected an unvisited vertex at random;
  2. Proceed from that vertex by using depth-first or breadth-first search, registering all visited vertices;
  3. Once the algorithm reaches a state in which there are no more reachable vertices, but all vertices were not still visited, it creates a subgraph containing all vertices visited and all edges connecting in those vertices;
  4. GOTO (1).
- **Connect Subgraphs** - As stated before, *RW* requires a connected graph. This can be achieved in a number of ways. The introduction of an edge between the two closest vertices of different subgraphs is chosen as the way to ensure connectedness. Note that the number of required additional edges to

ensure connectedness is given by the number of subgraphs minus 1. It was chosen however, to select each possible pair of subgraphs $\mathcal{G}_1$ and $\mathcal{G}_2$ and to insert an edge between $p_i \in \mathcal{G}_1$ and $p_j \in \mathcal{G}_2$ with the smallest distance among all such edges. The newly inserted edges receive weights in the same way as described in the next subsection. Given the constructed connected graph the *RW* consensus function previously described, it is applied for the ensemble clustering problems with no further modifications. Figure 4.3 shows an example in which a graph is generated with 3 disconnected subgraphs. The gray squares represent subgraphs, black lines edges between vertices (black circles), and red lines the edges to be added in order to make the graph connected.

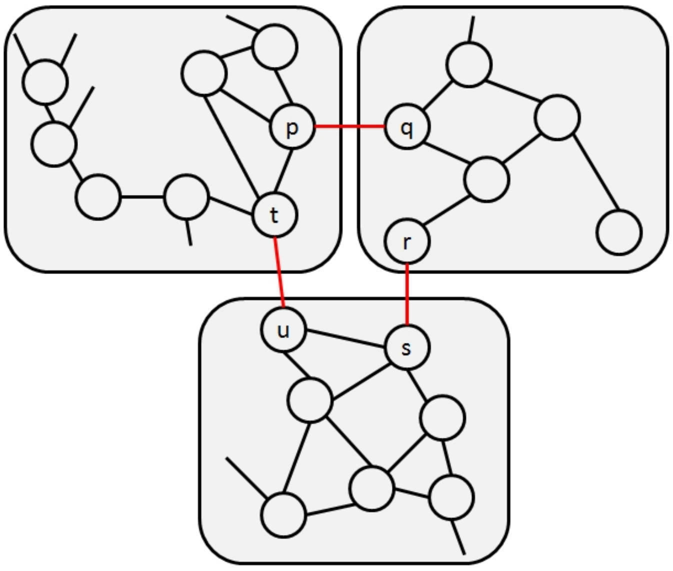


Figure 4.3. Example of a disconnected graph $\mathcal{G}$ composed of three subgraphs

By the end of this step, the graph $\mathcal{G}$ has the sets of vertices and edges fully defined. In order to finish the creation of the whole graph, the computation of the weights associated to each edge is in order.

Weight's Computation

The information contained in the clustering ensemble must be added to the graph. This is done by computing the weights assigned to each edge using as basis the information encoded in the ensemble of partitions.

The algorithm hereby proposed iterates over all vertices computing the edges weights. A weight $w_{i,j}$ indicates how probable two patterns p_i and p_j belong to the same cluster. The weighting method of choice is to count the number $m_{i,j}$ of initial partitions, in which p_i and p_j share the same region label. Thus, weight function is defined as a Gaussian weighting. The underlying assumption is based on the fact that objects belonging to the same "natural" region are very likely to be collocated in the same region among different partitions. The weighting function is defined as follows:

$$w_{i,j} = e^{-\beta \frac{m_{i,j}}{M}} \tag{4.2}$$

where $m_{i,j}$ refers to how many times the pair (i, j) of patterns occurs in the same cluster among the M partitions in the ensemble. The parameter β is a normalization parameter. For the purpose of the experiments presented later on this chapter it is set to 30.

Algorithm 4.1 implements the process of graph creation described above. The neighborhood size δ is specified as a parameter. The algorithm starts by computing the δ-nearest neighbors for each pattern.

Algorithm 4.1 δ-Neighborhood graph construction algorithm

```
Input: the original dataset $D$,
       the ensemble $\mathbb{P}$ of partitions to be combined,
       the neighborhood size $\delta$ to be considered
Output:a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$ representing the ensemble

1. For all $d \in D$
2.     find the $\delta$-Nearest Neighbors of $d$
3.     for all $nn \in \delta$-Nearest Neighbors($d$)
4.         compute the weight $w = weight(d, nn, \mathbb{P})$
5.         $\mathcal{E} = \mathcal{E} \cap e(d, nn, w)$
6.     end
7. end
8. if $\mathcal{G}$ is not connected
9.    find subgraphs
10.   connect subgraphs
11. end
```

An edge is created between each pattern and its δ-nearest neighbors. Sometimes two edges may be generated between two vertices p_i and p_j, one from considering p_i and the other one considering p_j. In this case, only one of them is added to $\mathcal{E}$. The next step deals with the weight computation. It computes a weight for each edge using Equation (4.2).

A naive implementation of the graph construction has $O(N(N\delta M))$ computational complexity. Note that M and $\delta << N$. However, the overall complexity can be improved by smarter ways to compute the δ nearest neighbors. For instance, the method proposed in [172] leads to a $O(N \log N)$ computational complexity implementation.

Once $\mathcal{G}$ is available, the computation of the consensual partition by means of the *RW* consensus function is in order. The original function can be used as stated before. The next section presents experimental results for a number of Synthetic and UCI-Irwine datasets.

4.3 Experimental Results

To assess the accuracy of the proposed method, four evaluation schemes are devised. First, the total number of nodes in the graph representation is investigated in respect to the size of the neighborhood considered. Afterwards, the impact of different neighborhood sizes in the overall performance of the method is evaluated. With a suitable neighborhood size defined, the results of *RW* are compared to other well stated ensemble clustering methods. Finally, the computational time required to execute the same methods evaluated in the last section is compared.

4.3.1 Assessing the Neighborhood Size

For this first experiment, the graph representation of the evaluated datasets is created using different neighborhood sizes.

The results are displayed in Table 4.2. It lists the number of patterns in each dataset, the number of produced edges using 4- and 8-neighborhoods and finally the number of edges for a complete graph representation. As it can be seen, the number of edges grows rapidly in relation to the neighborhood size considered. The progressive grown of the number of edges in relation to the neighborhood size can be observed in

Figure 4.4. It is clear that, the number of edges obeys a quadratic progressive increase. A small diminishment in the rate of increase of number of edges is observed once the neighborhood size increases. This is due to, more neighbors enter into play and the chance increases from two given patterns to share neighbors. However, the quadratic relation still holds.

Table 4.1. Impact of neighborhood size in the number of edges processed

Dataset	N.patterns	4-N	8-N	complete-N
iris	150	405	796	11175
wine	178	513	996	15753
breast	683	2234	4193	232903
optic	1003	2797	5469	499500
soyBeanS	47	118	221	1081
glass	218	612	1210	22791
haberman	306	816	1590	46665
mammo	830	2618	4527	344035
yeast	1484	4301	8431	1100386
halfRings	269	591	1197	36046
celipsoid	225	503	1001	25200
twoRings	362	851	1645	65341
scattered	132	342	656	8646

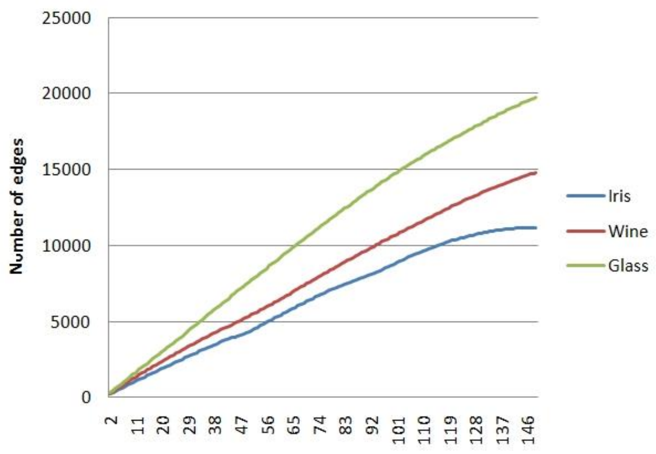


Figure 4.4. Grown of edges quantity *vs* neighborhood size

4.3.2 Neighborhood Size *vs* Accuracy

This test investigates the relationship between the neighborhood size used to create the graph and the error rate. If a small neighborhood can be chosen, a direct positive impact on the overall algorithm performance is observed, as discussed earlier.

Figure 4.5 shows the error rate obtained by the random walker algorithm with a known number of target clusters, for different neighborhood sizes. It is clear that the error rate archived by *RW* consensus function presents almost no fluctuation with increasing neighborhood sizes. In cases which very small neighborhoods are

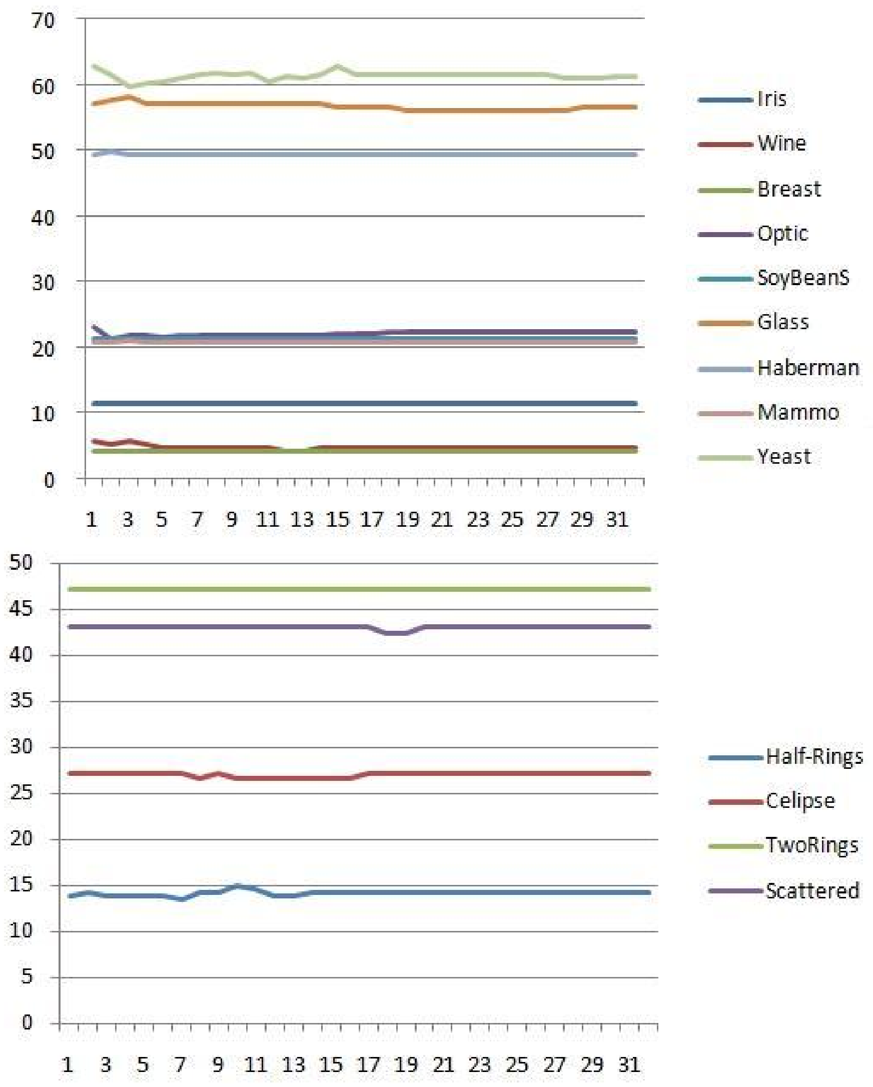


Figure 4.5. Relationship between error rate and neighborhood size

considered (e.g. up to 5 nearest neighbors) it is possible to observe a clear degradation in the algorithm's performance (refer e.g. to yeast dataset). This is due to the fact such neighborhood sizes do not incorporate enough information for the proper simulation of the random walkers. However, once the neighborhood size increases, little to non fluctuation is observed for the majority of the datasets. Exceptions are the yeast and wine datasets in which a slight decrease or increase in accuracy is observed with the increasing number of neighbors. This can occur due to the fact that parts of the clusters in those datasets overlap each other, causing different subsets of shared neighbors to change constantly.

For the majority of the inspected datasets a small neighborhood, like 6 closest neighbors lead to a result as good as larger counterparts such as e.g. 30 neighbors. This fact comes as a surprise since it is intuitively expected datasets with high attribute's dimensionality would require a larger neighborhood in order to create a representative graph. The general conclusion is that, a small neighborhood leading to similar results compared to a larger one, has a direct impact in the processing time required by the algorithm to achieve a consensual partition. However, the impact in performance should be more accentuated for larger datasets. This is the topic of the next envisioned experiment.

4.3.3 Assessing the Processing Time

Another interesting improvement is regarding the computational time required by the algorithm proposed. Table 4.2 shows that, for smaller data sets, *RW* consensus function has a matching performance compared to other ensemble clustering algorithms. Once the number of patterns in the dataset increases, a considerable improvement is observed. For the smallest dataset (soyBreanS - 47 patterns) *RW* processing time is actually worst than any other method compared. This is due to the fact that the graph creation step starts to pay off only once the number of patterns is relatively big. In some cases (e.g. yeast 1484 patterns) dataset, the time required by *RW* is much lower compared to the evidence accumulation methods. On the other hand, graph based methods tend to present a smaller computational time. This fact is balanced by the improvement in accuracy achieved by *RW*.

Table 4.2. Processing time for different ensemble clustering methods

Dataset	♯ patt.	*RW*	*EAC_SL*	*EAC_AL*	*HGPA*	*CSPA*
iris	150	0.06	0.03	0.05	0.09	0.14
wine	178	0.07	0.06	0.15	0.09	0.14
breast	683	1.75	12.12	44.88	0.10	1.20
optic	1000	4.18	16.02	20.93	0.14	1.27
soyBeanS	47	0.10	0.03	0.04	0.09	0.10
glass	214	0.13	0.08	0.16	0.11	0.21
haberman	306	0.54	0.28	0.76	0.09	0.41
mammo	830	0.54	0.28	0.76	0.09	0.41
yeast	1484	2.59	16.70	58.25	0.10	1.79

It is important to notice that the results given in Table 4.2 are produced using interpreted Matlab implementations of the compared algorithms. Those times can be greatly improved by using compiled versions algorithms. Additionally, one of the key impositions regarding the *RW* algorithm's computational complexity is regarding the computation of the nearest neighbors. Although seemly small neighborhoods are used, as discussed earlier, the computation of the nearest neighbors requires that the distance between all pairs or patterns to be computed and sorted. The times given in Table 4.2 are reported using a naive approach for this problem. More sophisticated approaches such as the one proposed in [172] can lead to even greater performance improvements.

4.3.4 Assessing the Overall Performance

To evaluate the accuracy of the consensus partitions obtained by the *RW* consensus function a comparison to well known consensus functions widely used throughout the pertinent literature is envisioned. The methods of choice cover the main consensus function classes presented in Chapter 3. *BoK* and $BOEM$ [67] are selected to represent the median partition formulation. *BoK* is essentially the set median as stated before, and $BOEM$ does not require the number of clusters to be specified a priori. The result obtained by *BoK* is provided to $BOEM$ as initial consensus partition. Similarly, EAC_SL and EAC_SL [57] are selected to represent voting methods. $HGPA$ and $CSPA$ [140] represent the graph based methods in which both require the number of desired clusters to be known. The results for the random walker algorithm are presented in two versions. *RW* stands for the original version able to decide the optimal number of clusters and $RWfix$ which the number of clusters is known. For all results presented, those algorithms requiring a known K used the number of target clusters extracted from the available ground-truths.

The ensembles used in order to produce the results follow the ensemble generation schemes described in Chapter 2. For each of the 40 generation schemes, 10 ensembles are computed with number of partitions varying from 10 to 40. The results presented in the remainder of this chapter are the average of all 400 ensembles computed for each dataset.

Table 4.3 shows the results computed using the variation of information index for the six toy datasets described in Chapter 2. For these datasets, no remarkable improvement can be noticed except for the celipsiod dataset. However it is interesting to see that neither *RW* nor $RWfix$ performed worst in any case. The importance of this observation is that, despite of the fact *RW* can use a priori knowledge about the number of desired clusters this is not, by any means a requirement, defining it as a single consensus solution. The same does not apply to neither of the graph based methods nor the voting methods.

Table 4.3. VI index for the toy datasets

Dataset	*BoK*	*BOEM*	*EAC_SL*	*EAC_A*L	*HGPA*	*CSPA*	*RW*	*RWfix*
8D5K	0.04	0.02	0.00	0.00	4.55	0.00	0.02	0.02
2D2K	0.27	0.27	0.27	0.27	2.00	0.32	0.27	0.27
celipsoid	1.68	1.68	1.68	1.68	1.99	1.67	1.69	1.52
twoRings	1.97	1.97	1.97	1.97	1.99	1.99	1.97	1.97
scattered	1.84	1.84	1.84	1.84	1.86	1.86	1.84	1.84
halfrings	1.14	1.14	1.14	1.14	1.97	1.29	1.14	1.14

Table 4.4 shows the error ratio or, the number of wrongly classified patterns for the same toy datasets. Here, the results do not change much, compared to Table 4.3. However, for celipsoid, it becomes clearer when the results for $HGPA$ and $CSPA$ are compared that the lack of a single solution can introduce doubts about which consensus function to use when a single solution such as *RW* is not available.

Table 4.4. Error rates (in %) for the toy datasets

Dataset	*BoK*	*BOEM*	*EAC_SL*	*EAC_A*L	*HGPA*	*CSPA*	*RW*	*RWfix*
8D5K	0.20	0.10	0.00	0.00	77.10	0.00	0.10	0.10
2D2K	1.90	1.90	1.90	1.90	49.40	2.30	1.90	1.90
celipsoid	27.56	27.56	27.56	27.56	49.78	27.11	28.00	26.02
twoRings	47.51	47.51	47.51	47.51	48.62	48.90	47.51	47.51
scattered	43.18	43.18	43.18	43.18	48.48	46.97	43.18	43.18
halfrings	13.75	13.75	13.75	13.75	49.81	17.84	13.75	13.75

Table 4.5 presents the results using VI index for the selected 25 UCI-Irvine datasets. For this evaluation, it can be seen tangible improvements introduced by the *RW* consensus function. In 21 of the 25 datasets, *RW* perform better or at least similar to the best result of any other algorithm. For the datasets balance, breast, glass, haberman, optic, post-op, protein, soybean, transfusion, wine and yeast, *RW* is clearly superior to any compared algorithm showing remarkable improvements. Exceptions are the ecoli dataset, in which case the *EAC_SL* achieves a better result. But, it is remarkable to notice that *EAC_SL* is one of the possible evidence accumulation methods. For the same dataset, the *RW* consensus function is capable of deciding automatically the optimal number of clusters achieving good results well in between the two evidence accumulation methods proving the reliance of the random walker formulation. For spect and transfusion datasets *EAC_SL* also achieves better results, but the difference when compared to *RW* is negligible. The segmentation dataset is the only case in which a graph based method ($CSPA$) succeeds in achieving a better score than any other method. For the remainder datasets,

RW achieves at least as well as any other method with a slightly variation in either direction.

Table 4.5. VI index for the UCI-Irvine datasets

Dataset	*BoK*	*BOEM*	*EAC_SL*	*EAC_A*L	*HGPA*	*CSPA*	*RW*	*RWfix*
balance	2.25	2.25	2.63	2.52	2.77	2.70	1.66	2.52
Breast	0.47	0.47	0.95	0.46	1.93	1.13	0.40	0.40
control	1.51	1.50	1.36	1.48	2.41	1.31	1.30	1.60
ecoli	2.33	2.34	1.82	2.23	3.28	2.91	2.10	2.36
glass	2.31	2.34	1.90	1.99	3.08	3.38	1.33	1.27
haberman	1.28	1.28	1.28	1.28	1.83	1.83	0.83	0.91
heart	1.86	1.86	1.86	1.86	1.99	1.91	1.86	1.86
ionosphere	1.73	1.73	1.73	1.73	1.90	1.77	1.74	1.74
iris	0.43	0.43	0.73	0.49	2.15	0.54	0.13	0.17
lung	1.47	1.47	1.42	1.42	1.55	1.54	1.41	1.44
mammo	1.42	1.42	1.42	1.42	2.00	1.51	1.44	1.44
optic	1.56	1.58	2.09	1.70	3.73	1.66	0.77	0.56
parkinsons	1.25	1.25	1.25	1.25	1.80	1.74	1.25	1.25
post-op	2.01	2.01	1.96	1.98	2.45	2.47	1.72	1.94
protein	1.54	1.54	1.56	1.57	3.24	2.89	1.64	1.43
segmentation	1.82	1.82	1.82	1.82	2.34	1.74	1.97	2.05
sonar	1.99	1.99	1.94	1.95	1.99	1.97	1.96	1.94
soyBeanS	1.28	1.28	1.28	1.28	1.28	1.67	0.51	1.07
spect	1.58	1.58	1.46	1.55	1.73	1.57	1.55	1.54
spectf	1.25	1.25	1.24	1.24	1.73	1.60	1.23	1.23
taeval	2.91	2.91	2.91	2.91	3.04	2.97	2.87	2.87
tic-tac-toe	2.08	2.08	1.47	1.87	1.93	1.91	1.47	1.87
transfusion	1.41	1.41	1.38	1.40	1.79	1.76	1.40	1.40
wine	0.38	0.38	0.34	0.34	1.58	0.56	0.19	0.19
yeast	3.63	3.72	2.53	3.53	5.29	4.63	1.88	1.45

Table 4.6 shows the error ratio or, the number of wrongly classified patterns for the same UCI-Irvine datasets. For this index again, *RW* presents better results for most datasets (balance, breast, glass, haberman, optic, post-op, protein, soybean, spectf, wine and yeast). Exceptions are the control, segmentation, and spect. For these datasets, *CSPA* shows a better performance. However, *RW* again shows a mid value performance between *CSPA* and *HGPA*, the two graph based methods proving once more the reliance of the random walker formulation. The same applies to ecoli dataset in which case *EAC_SL* shows a better result, however *EAC_AL*, the other evidence accumulation method evaluated is clearly inferior. For the remainder

datasets, *RW* achieves at least as well as any other method with a slightly variation in either direction.

Table 4.6. Error rates (in %) for the UCI-Irvine datasets

Dataset	*BoK*	*BOEM*	*EAC_SL*	*EAC_AL*	*HGPA*	*CSPA*	*RW*	*RWfix*
balance	39.68	39.68	33.28	39.68	57.28	46.08	29.60	39.68
breast	3.95	3.95	34.85	3.81	49.63	17.28	3.22	3.22
control	39.58	39.68	39.34	37.70	44.16	21.71	47.40	38.76
ecoli	51.61	51.58	34.17	44.60	64.78	58.12	40.34	46.58
glass	46.26	46.26	48.13	48.60	58.88	64.49	44.86	42.52
haberman	25.16	25.16	25.16	25.16	49.67	48.04	26.47	22.88
heart	38.28	38.28	38.28	38.28	47.65	39.16	38.30	38.30
ionosphere	30.68	30.67	30.68	30.68	42.81	33.95	31.11	31.11
iris	5.25	5.12	32.00	4.67	35.00	5.33	4.00	4.67
lung	29.07	29.07	26.94	26.30	30.46	29.54	26.76	27.13
mammo	20.72	20.72	20.72	20.72	48.19	21.69	21.08	21.08
optic	20.70	20.90	54.00	27.20	57.40	17.50	21.50	11.90
parkinsons	27.88	27.88	27.88	27.88	47.76	41.44	27.76	27.76
post-op	46.41	46.41	43.53	43.76	61.38	63.28	40.26	43.33
protein	50.82	50.82	50.69	50.75	56.40	51.81	55.00	48.06
segmentation	43.81	43.81	43.81	43.81	39.05	28.57	44.29	42.38
sonar	45.43	45.44	43.98	44.10	48.34	44.17	44.59	44.33
soyBeanS	29.79	29.79	29.79	29.79	29.79	34.04	19.15	27.66
spect	43.71	43.62	39.63	43.38	49.42	37.55	43.95	44.09
spectf	36.63	36.63	36.13	35.98	49.41	39.40	35.51	35.51
taeval	52.98	52.98	52.98	52.98	60.26	50.33	53.64	53.64
tic-tac-toe	51.35	51.21	39.62	46.54	50.00	45.59	39.60	46.54
transfusion	32.77	32.77	30.87	31.09	48.36	45.01	31.03	31.01
wine	3.37	3.37	2.81	2.81	47.75	6.18	1.69	1.69
yeast	60.31	60.78	68.40	53.30	79.18	72.37	42.25	34.64

The random walker consensus function adapted to work with general datasets has proven to be a suitable choice to address the problem of ensemble clustering. Due to the fact that most general datasets are comprised of patterns with a high number of dimensions (usually much higher than 2 or 3) they do not present a well behaved lattice. The datasets high dimensionality complicate the definition of a rigid neighborhood structure. A neighborhood generation method is therefore, required. The computation of the δ-closest neighbors presents itself as a plausible option. Nevertheless, the definition of a suitable values for δ is an open issue. As demonstrated in Section 4.3.1 a conceivably small neighborhood size can be chosen with no considerable loss in performance, since the achieved results fluctuate very little for increasing δ values. This fact alone enables the possibility of applying the proposed consensus functions to medium to large datasets. Several other advantages are observed in comparing RW to other state of art consensus functions. Although, using an evidence accumulation scheme similar to the one adopted by EAC_AL/EAC_SL, differently from those methods, there is no need to compute a complete co-association matrix, but only the weights associated to the edges in the graph. This fact improves even further the computational time. The final combination step uses a random walker consensus function that can be solved exactly, without the simulation of random walks, by solving a sparse symmetric positive-definite system of equations. This method is remarkably quicker compared to the hierarchical clustering algorithms used in the consensus step of EAC_AL/EAC_SL algorithms.

Regarding computational time, RW does not present a representative gain if compared to the graph-based methods. However, by comparing the accuracy of both methods, RW clearly outperforms the other graph based methods.

Chapter 5

Lower Bound for Ensemble Clustering

Cluster ensemble techniques are means for boosting the clustering performance. However, many cluster ensemble methods are faced with high computational complexity. Indeed, the median partition methods are $\mathcal{NP}$-complete for many reasonable distance functions. One essential aspect in this context is the assessment of the quality of the computed approximate solutions. While a variety of approximate approaches for suboptimal solutions has been proposed in the literature as shown in Chapter 3, the performance evaluation is typically done by means of ground-truth (see Chapter 2). In contrast, this chapter explores the question how well the cluster ensemble methods perform in an absolute sense *without ground-truth*, i.e. how they compare to the (unknown) optimal solution.

A study is presented of applying and extending a lower bound Γ as an attempt to investigate this question. In particular, it demonstrates the tightness of the lower bound, which indicates that there exists no more room for further improvement (for the particular data set at hand). Thus, the lower bound can be considered as means of exploring the performance limit of cluster ensemble techniques.

The remainder of this chapter is organized as follows. Section 5.1 describes the lower bound based on linear programming applied to clustering ensemble. Section 5.2 presents a study of the lower bound Γ using three clustering ensemble methods and eleven data sets. Among others it will be demonstrated that this lower bound can (almost) be reached by the computed solution. This tightness indicates the limited room for further improvement. Therefore, the lower bound Γ represents a means of exploring the performance limit of cluster ensemble techniques. Section 5.3

presents other related lower bounds. Section 5.4 extends the lower bound to deal with weighted cluster ensemble techniques. Finally, the chapter ends in Section 5.5 with conclusions and final remarks.

5.1 LP-Based Lower Bound

Order theory defines the lower bound Γ [33] of a subset S of an ordered set $(P, \leq)$ is an element of P which is smaller than or equal to every element of S. A set with a lower bound is said to be bounded from below by Γ.

Given the data set $X = \{x_1, x_2, \ldots, x_n\}$ of n patterns x_i, a cluster ensemble is a set $\mathbb{P} = \{P_1, P_2, \ldots, P_M\}$ of M partitions P_i. $P_i \in \mathbb{P}$ is a clustering of X. The set of all possible clusterings of X is denoted by $\mathcal{P}_X$ ($\mathbb{P} \subset \mathcal{P}_X$). The goal of cluster ensemble techniques is to find a consensus clustering $P^* \in \mathcal{P}_X$, which optimally represents the ensemble $\mathbb{P}$.

In median partition methods this optimality is formulated as:

$$P^* = \arg \min_{P \in \mathcal{P}_X} \sum_{i=1}^{M} d(P, P_i) \tag{5.1}$$

where $d(\cdot, \cdot)$ is a distance (dissimilarity) function between two clusterings. Note that this definition is a special instance of the so-called generalized median problem, which has been intensively investigated in structural pattern recognition, see [78, 103] for the case of strings and graphs.

The median partition problem has been proven to be $\mathcal{NP}$-complete [10] for many reasonable distance functions. An exhaustive search in $\mathcal{P}_X$ is computationally intractable. In practice suboptimal approaches [105, 140] are thus developed to solve the optimization problem.

Given a suboptimal solution $\tilde{P} \in \mathcal{P}_X$, the question of its accuracy arises. In [79] a lower bound is proposed to answer this question (for the general case of generalized median problems). For an approximate solution $\tilde{P}$ the following relationship holds:

$$SoD(\tilde{P}) = \sum_{i=1}^{M} d(\tilde{P}, P_i) \geq \sum_{i=1}^{M} d(P^*, P_i) = SoD(P^*) \tag{5.2}$$

where SoD stands for sum of distances. The quality of $\tilde{P}$ can be absolutely measured by the difference $SoD(\tilde{P}) - SoD(P^*)$. Since P^* and $SoD(P^*)$ are unknown in general, one can resort to a lower bound Γ:

$$0 \le \Gamma \le SoD(P^*) \le SoD(\tilde{P}) \tag{5.3}$$

to measure the quality of $\tilde{P}$ by $SoD(\tilde{P}) - \Gamma$ instead. Obviously, the trivial lower bound $\Gamma = 0$ is useless. The lower bound Γ is required to be as close to $SoD(P^*)$ as possible.

In [79] a lower bound based on linear programming is proposed for metric spaces. Assuming a metric distance function $d(\cdot,\cdot)$, the lower bound for the median partition problem is specified by the solution Γ of the following linear program:

$$\begin{aligned} &\text{minimize } x_1 + x_2 + \cdots + x_M \text{ subject to} \\ &\forall\, i,j \in \{1,2,\ldots,M\},\ i \ne j, \begin{cases} x_i + x_j & \ge\ d(P_i,P_j) \\ x_i + d(P_i,P_j) & \ge\ x_j \\ x_j + d(P_i,P_j) & \ge\ x_i \end{cases} \\ &\forall\, i \in \{1,2,\ldots,M\},\ x_i \ge 0 \end{aligned} \tag{5.4}$$

Given a suboptimal solution $\tilde{P}$ and the computed lower bound, the deviation $\Delta = SoD(\tilde{P}) - \Gamma$ can give a hint of the absolute accuracy of $\tilde{P}$. In particular, if $\Delta \approx 0$, then it can be safely claimed that there is hardly room for further improvement (for the particular data set at hand).

5.2 Experimental Verification

In order to assess the usefulness of the LP-lower bound applied to the clustering ensemble problem, a comparison study is presented using two cluster ensemble methods and eleven data sets. Among others it will be demonstrated that this lower bound can (almost) be reached by the computed solution. This tightness indicates the limited room for further improvement. Therefore, the lower bound Γ represents means of exploring the performance limit of cluster ensemble techniques.

The lower bound proposed is tested against three ensemble clustering methods. Given an ensemble $\mathbb{P}$, a final clustering $\tilde{P}$ is computed using either *EAC_AL*, *EAC_SL* or *RW*.

$$\Delta' = \frac{SoD(\tilde{P}) - \Gamma}{SoD(\tilde{P})} \tag{5.5}$$

The following measures are used to characterize the performance: $SoD(\tilde{P})$, the lower bound Γ (for the ensemble), and the deviation (in percentage). For each data set, this procedure is repeated ten times (i.e. ten different ensembles) and the average measures are reported.

The performance measures for the three cluster ensemble methods, RW, EAC_AL and RW, EAC_SL are presented in Table 5.1, 5.3 and 5.2, respectively.

Table 5.1. Deviation Δ' for the RW method

	d_{vi}			d_{vd}			d_m		
Dataset	$SoD(\tilde{P})$	Γ	$\Delta'(\%)$	$SoD(\tilde{P})$	Γ	$\Delta'(\%)$	$SoD(\tilde{P})$	Γ	$\Delta'(\%)$
iris	8.40	7.24	13.8	2.28	2.16	5.2	28067	25113	10.5
wine	2.09	1.86	10.0	0.35	0.33	4.5	7242	6777	5.8
breast	1.49	1.08	27.7	0.20	0.15	23.9	90032	68392	24.0
optic	11.38	6.37	44.0	3.90	1.85	50.9	749459	315016	57.7
soyBeanS	6.19	3.79	36.9	4.08	1.62	52.0	3433	1591	49.3
glass	7.96	4.66	41.1	2.53	1.24	45.9	69186	33940	49.3
haberman	7.70	7.58	1.5	2.86	2.84	0.7	234484	232995	0.6
mammo	1.77	1.77	0.0	0.38	0.38	0.0	248650	248650	0.0
yeast	18.60	11.40	38.2	10.51	3.34	67.5	6606869	3010185	53.4
2D2K	4.69	4.69	0.0	1.15	1.15	0.0	978050	978050	0.0
8D5K	5.24	4.91	5.9	2.43	1.66	15.0	721412	579262	11.3

The deviation P' can be interpreted as the potential of further improvement. For three data sets (haberman, mammo, and 2D2K) $SoD(\tilde{P})$ almost reaches the lower bound Γ for all three distance functions, indicating practically no room for improvement. To some extent the same applies to the data set Soy and 8D5K in conjunction with EAC_AL.

In these cases the lower bound turns out to be extremely tight. On the other hand, if the deviation is large, care must be taken in making any claims. The large deviation may be caused by two reasons: The lower bound is not tight enough in that particular case or the computed solution $SoD(\tilde{P})$ is still far away from the (unknown) optimal solution P^*. The second case is certainly more delicate. But it may be interpret as of some, although uncertain, potential of further improvement. Given such an ensemble, we could generate more ensembles and compute additional candidates for consensus clustering. The measure SoD can then be used for selecting a final solution. This strategy has been suggested in [140] (although in a different context): "*The objective function has the added advantage that it allows one to add a stage that selects the best consensus function without any supervisory information,*

Table 5.2. Deviation Δ' for the *EAC_SL* method

	d_{vi}			d_{vd}			d_m		
Dataset	$SoD(\tilde{P})$	Γ	$\Delta'(\%)$	$SoD(\tilde{P})$	Γ	$\Delta'(\%)$	$SoD(\tilde{P})$	Γ	$\Delta'(\%)$
iris	9.33	7.24	0.22	3.23	2.16	0.33	48991	25113	0.49
wine	1.95	1.86	0.05	0.33	0.33	0.00	6881	6776	0.02
breast	9.33	1.08	0.88	3.36	0.15	0.96	2078788	68392	0.97
optic	14.52	6.37	0.56	4.17	1.85	0.56	2343771	315016	0.87
soyBeanS	3.90	3.79	0.03	1.65	1.62	0.02	1616.60	1591	0.02
glass	6.22	4.66	0.25	1.73	1.24	0.28	44219.80	33939	0.23
haberman	7.58	7.58	0.00	2.88	2.84	0.01	233456	232994	0.00
mammo	7.55	1.77	0.77	2.48	0.38	0.85	679674	248649	0.63
yeast	20.60	11.40	0.45	6.02	3.34	0.45	14864622	3010184	0.80
8D5K	5.72	4.91	0.14	2.09	1.66	0.21	754567	579262	0.23
2D2K	4.72	4.69	0.01	1.33	1.15	0.14	1190372	978049	0.18

by simply selecting the one with the highest ANMI" (ANMI is the particular *SoD* used in that work). Thus, a tight lower bound may give a hint to continue or terminate the procedure without any knowledge of ground-truth.

Table 5.3. Deviation Δ' for the *EAC_AL* method

	d_{vi}			d_{vd}			d_m		
Dataset	$SoD(\tilde{P})$	Γ	$\Delta'(\%)$	$SoD(\tilde{P})$	Γ	$\Delta'(\%)$	$SoD(\tilde{P})$	Γ	$\Delta'(\%)$
iris	8.22	7.24	12.0	2.26	2.16	4.3	27621	25113	9.1
wine	2.01	1.86	7.7	0.35	0.33	5.1	7232	6777	6.3
breast	1.16	1.08	7.3	0.16	0.15	3.8	71244	68392	4.0
optic	7.50	6.37	15.0	2.06	1.85	10.0	378439	315016	16.8
soyBeanS	3.90	3.79	2.9	1.65	1.62	1.9	1616	1591	1.6
glass	5.20	4.66	10.4	1.37	1.24	9.4	39909	33939	15.8
haberman	7.60	7.58	0.3	2.84	2.84	0.0	233417	232994	0.2
mammo	1.77	1.77	0.0	0.38	0.38	0.0	248649	248649	0.0
yeast	13.94	11.40	18.3	3.85	3.34	13.4	3512666	3010184	14.3
2D2K	4.86	4.69	3.0	1.18	1.15	3.0	1037580	978050	5.7
8D5K	4.97	4.91	1.8	1.69	1.66	2.0	585462	579262	1.1

There is also the issue of inconsistency among different distance functions. Sometimes it happens that the deviation values for two distance functions vary, partly substantially. This observation is not really surprising. Different distance functions may not share the same view of dissimilarity, thus the quality of a consensus clustering. It is up to the user to decide which distance function is more suitable for a particular data clustering task.

Finally, it is relevant to point out that the three cluster ensembles methods used in this study do not belong to the class of median partition techniques. But even in this case the lower bound still provides useful information about the optimality of the computed consensus clustering.

5.3 Other Ensemble Clustering Lower Bounds

The cluster ensemble problem with Merkin distance d_m has been intensively investigated [64, 67]. This is mainly due to the simplicity of d_m, which allows to obtain deep insight into this particular consensus clustering problem. In particular, several suboptimal algorithms have been proposed with known approximation factor. In addition, a lower bound specific to d_m only can be defined:

$$\Gamma_m = \sum_{i<j} \min\Big(\sum_{k=1}^{M} X_{ij}^{(k)},\ N - \sum_{k=1}^{M} X_{ij}^{(k)}\Big) \tag{5.6}$$

where $X_{ij}^{(k)}$ is the Bernoulli random variable as 1 if x_i and x_j are co-clustered in partition P_k and 0 otherwise. Γ_m takes the specific properties of d_m into account, whereas Γ is based on the general properties of a metric only. Γ_m is better informed and expected to be tighter than Γ. Table 5.4 compares the closeness of the two lower bounds. It is remarkable that without any knowledge of d_m and using the metric properties alone, the general lower bound Γ almost reaches Γ_m.

Table 5.4. Comparison of lower bounds Γ and Γ_m

Dataset	Γ	Γ_m	$(\Gamma_m - \Gamma)/\Gamma(\%)$
iris	25113	26377	5.0
wine	6777	6820	0.6
breast	68392	71196	4.1
optic	315016	335678	6.6
soyBeanS	1591	1599	0.5
glass	33940	34513	1.7
haberman	232995	233273	0.1
mammo	248650	248650	0.0
yeast	3010185	3224160	7.1
2D2K	978050	1168728	8.4
8D5K	579262	584848	1.0

Another interesting lower bound, originally developed in the context of median graphs, was proposed by Jiang *et al.* [78]. It is possible to adapt it for general median problems. It also requires the distance between objects to be a metric. Additionally, the set of objects $\mathbb{P} = \{P_1, P_2, \cdots, P_M\}$ must have an even number of objects. In this case, the true generalized median $\bar{P}$ is subject to:

$$\begin{aligned} SoD(\bar{P}) &= [d(\bar{P}, P_1) + d(\bar{P}, P_2)] + [d(\bar{P}, P_3) + d(\bar{P}, P_4)] + \cdots \\ &\quad + [d(\bar{P}, P_{M-1}) + d(\bar{P}, P_M)] \\ &\geq d(P_1, P_2) + d(P_3, P_4) + \cdots + d(P_{M-1}, P_M) \end{aligned} \tag{5.7}$$

This relationship remains true for any partition of $\mathbb{P}$ into $M/2$ pairs. There are $\frac{M!}{2^{\frac{M}{2}} \cdot (\frac{M}{2})!}$ possible ways to partition the set $\mathbb{P}$ into:

$$(P_{l,1}, P_{l,2}), (P_{l,3}, P_{l,4}), \cdots, (P_{l,M-1}, P_{l,M}) \tag{5.8}$$

where $\{P_{l,1}, P_{l,2}, \cdots, P_{l,M}\} = \mathbb{P}$. Therefore, provided $d(\cdot,\cdot)$ is a metric, the true generalized median $\bar{P}$ satisfies

$$\begin{aligned} SoD(\bar{P}) \geq \; & \max\{d(P_{l,1}, P_{l,2}) + d(P_{l,3}, P_{l,4}) + \cdots + d(P_{l,M-1}, P_{l,M}) \\ & |((P_{l,1}, P_{l,2}), (P_{l,3}, P_{l,4}), \cdots, (P_{l,M-1}, P_{l,M})) \text{ is a partition of } \mathbb{P}\} \end{aligned} \tag{5.9}$$

It is observed experimentally that this pairwise lower bound and the one based on linear programming turned out to consistently have the same value. However, the mathematical proof of a conjectured equivalence is still under investigation.

5.4 Extension to Weighted Cluster Ensemble Techniques

Cluster ensembles techniques can be extended by assigning a weight w_i to each involved partition P_i, which represents the estimated relative merit of the partitions. For instance, in [154] four weights are considered: a) inter-cluster distance, b) intra-cluster distance, c) mean size of clusters, and d) difference between the cluster sizes. In order to account for the weights associated to each partition, the weighted median partition problem can be adjusted as:

$$P^* = \arg \min_{P \in \mathcal{P}_X} \sum_{P_i \in \mathbb{P}} w_i \cdot d(P, P_i) \tag{5.10}$$

Table 5.5. Deviation Δ' for the weighted version of *RW* method

	d_m			d_{vd}			d_{vi}		
Dataset	$SoD(\tilde{P})$	Γ_w	$\Delta'(\%)$	$SoD(\tilde{P})$	Γ_w	$\Delta'(\%)$	$SoD(\tilde{P})$	Γ_w	$\Delta'(\%)$
iris	0.81	0.68	16.0	0.22	0.20	9.1	2753	2356	14.4
wine	0.64	0.19	70.8	0.11	0.03	70.8	2303	677	70.6
breast	0.22	0.11	50.6	0.03	0.02	50.5	13819	6834	50.5
optic	1.12	0.64	43.0	0.36	0.19	46.0	55409	31492	42.2
soyBeanS	0.52	0.38	25.5	0.39	0.16	47.6	307	157	42.3
glass	0.85	0.47	44.7	0.30	0.13	51.9	6436	3422	42.5
haberman	0.80	0.76	4.3	0.29	0.29	1.4	24101	23303	3.3
mammo	0.17	0.17	0.0	0.04	0.04	0.0	23794	23794	0.0
yeast	1.85	1.14	38.8	1.02	0.33	66.7	511552	299571	40.8
2D2K	0.52	0.52	0.9	0.13	0.13	0.9	108495	107833	0.5
8D5K	0.52	0.48	5.8	0.24	0.16	15.0	70603	56218	11.3

The extension of the linear program lower bound Γ to deal with the weighted cluster ensemble problem is straightforward, resulting in a lower bound Γ_w. It is formulated as follows.

$$\text{minimize } w_1 \cdot x_1 + w_2 \cdot x_2 + \cdots + w_M \cdot x_M \text{ subject to}$$

$$\begin{array}{l} \forall i,j \in \{1,2,\ldots,M\},\ i \neq j, \left\{ \begin{array}{rcl} x_i + x_j & \geq & d(P_i,P_j) \\ x_i + d(P_i,P_j) & \geq & x_j \\ x_j + d(P_i,P_j) & \geq & x_i \end{array} \right. \\ \forall i \in \{1,2,\ldots,M\},\ x_i \geq 0 \end{array} \tag{5.11}$$

Many cluster ensembles methods can be easily extended to integrate such weights. In co-occurrence based techniques such as *EAC_AL* and *RW* this can be done when computing the co-occurrence matrix. For the experiment presented here, the inter-cluster distance is used.

For these weighted algorithms the performance measures are shown in Table 5.5, 5.7, and 5.6. Compared to the unweighted results the things have not changed much.

For the three data sets haberman, mammo, and 2D2K, $SoD(\tilde{P})$ again almost reach the lower bound Γ_w for all three distance functions, indicating practically no room for further improvement. In conjunction with *EAC_AL*. The same can be said about the data set 8D5K. In these cases the lower bound turns out to be extremely tight. On the other hand, if the deviation is larger, one must be careful

Table 5.6. Deviation Δ' for the weighted version of (EAC_SL) method

	d_m			d_{vd}			d_{vi}		
dataset	$SoD(\tilde{P})$	Γ_w	$\Delta'(\%)$	$SoD(\tilde{P})$	Γ_w	$\Delta'(\%)$	$SoD(\tilde{P})$	Γ_w	$\Delta'(\%)$
iris	0.82	0.68	0.16	0.26	0.20	0.17	3511	2356	0.24
wine	0.20	0.19	0.07	0.03	0.03	0.03	711	678	0.05
breast	0.12	0.11	0.07	0.02	0.01	0.04	7119	6834	0.04
optic	1.38	0.64	0.51	0.40	0.18	0.52	216109	31492	0.80
soyBeanS	0.39	0.38	0.02	0.16	0.16	0.01	160	158	0.01
glass	0.63	0.47	0.25	0.18	0.12	0.27	4416	3423	0.21
haberman	0.77	0.76	0.01	0.29	0.29	0.01	23754	23303	0.02
mammo	0.17	0.17	0.00	0.04	0.04	0.00	23794	23794	0.00
yeast	2.07	1.14	0.45	0.60	0.33	0.45	1485159	299571	0.80
8D5K	0.53	0.48	0.07	0.19	0.16	0.09	66239	56219	0.10
2D2K	0.51	0.51	0.00	0.13	0.13	0.00	107322	107834	0.00

Table 5.7. Deviation Δ' for the weighted version of (EAC_AL) method

	d_m			d_{vd}			d_{vi}		
Dataset	$SoD(\tilde{P})$	Γ_w	$\Delta'(\%)$	$SoD(\tilde{P})$	Γ_w	$\Delta'(\%)$	$SoD(\tilde{P})$	Γ_w	$\Delta'(\%)$
iris	0.78	0.68	12.1	0.21	0.12	4.5	2599	2356	9.2
wine	0.20	0.19	7.5	0.04	0.03	5.0	723	678	6.2
breast	0.12	0.11	7.3	0.02	0.02	3.8	7119	6834	4.0
optic	0.75	0.64	14.7	0.21	0.19	9.7	36742	31492	13.9
soyBeanS	0.39	0.38	2.2	0.16	0.16	1.4	160	158	1.2
glass	0.52	0.47	10.5	0.14	0.12	9.6	3996	3423	12.5
haberman	0.77	0.76	1.5	0.29	0.29	0.8	23754	23303	1.9
mammo	0.17	0.17	0.0	0.04	0.04	0.0	23794	23794	0.0
yeast	1.40	1.14	18.4	0.38	0.33	13.2	353189	299571	15.0
2D2K	0.52	0.52	0.0	0.13	0.13	0.0	107322	107834	0.0
8D5K	0.49	0.48	1.3	0.16	0.16	1.6	56825	56218	1.0

in making any claims. Also here, the deviation can be seen as a hint for continuing optimization.

5.5 Conclusions

This chapter presented a study of the lower bound Γ using eleven data sets. It could be shown:

- In some cases this lower bound can (almost) be reached by the computed solution. This tightness implies that there exists no more room for further improvement for this particular data set (with respect to the used distance function). Larger deviation may indicate some, although uncertain, potential of improvement and thus serves as a hint for continuing optimization.
- The same observation can be made also for weighted version of cluster ensemble methods.
- The tightness of Γ can be even demonstrated in case of Merkin distance d_m by comparing with another lower bound, which is derived from the special nature of d_m.

Based on these facts the lower bound Γ (and Γ_m are considered in case of d_m) as means of exploring the performance limit of cluster ensemble techniques.

The lower bound defined in [79] presumes a metric distance function $d(\cdot,\cdot)$. The triangle inequality of a metric excludes cases in which $d(P,R)$ and $d(R,Q)$ are both small, but $d(P,Q)$ is very large. In practice, however, there may exist distance functions which do not satisfy the triangle inequality. The work [47] extends the concept of metrics to a relaxed triangle inequality. Instead of the strict triangle inequality, the relation:

$$d(P,R) + d(R,Q) \geq \frac{d(P,Q)}{1+\varepsilon} \tag{5.12}$$

is required, where ε is a small nonnegative constant. This is also called quasi-metric in mathematics [71]. As long as ε is not very large, the relaxed triangle inequality still retains the human intuition of similarity. Note that the strict triangle inequality is a special case with $\varepsilon = 0$. The lower bound Γ can be easily extended to quasi-metric distance functions by changing the inequalities in the linear program accordingly. This extended lower bound can be expected to be useful in working with cluster ensemble methods based on quasi-metrics.

Since the lower bound is computed based on the ensemble available these questions remains: If the result obtained by a given consensus function succeeds in approximating the lower bound computed this is indeed a good solution? and similarly: If the result obtained by a given consensus function fails in approximating the lower bound it is correct to assume that the consensus function used is not a good choice?

In fact, what the comparison against the lower bound proposed tells is that for a given consensus function and for the provided ensemble those conditions holds. Different ensemble generation schemes such as the ones proposed in Chapter 3 will have a direct impact in the lower bound, being in fact different for each ensemble. Similarly, the consensus partition possible to be obtained will also be dependant of the ensemble of partitions provided. Considering the case in which no ground-truth is provided and no knowledge about the data distribution is available, one can only hope to estimate the accuracy of the consensus partition produced. This is exactly what the lower bound proposed provides, a means of access the quality of the result obtained in respect to the ensemble provided and the consensus partition adopted.

Chapter 6

Image Segmentation Combination via Ensemble Clustering

Image segmentation is the first step and also, one of the most critical tasks in image analysis. In order to deal with the great variability of features encountered in different images, specific segmentation methods have been designed for different types of images. Examples are, medical [143], range [73], and outdoor images [151, 152] among many others. Considering a single image type, as e.g., outdoor images presented in Figure 6.1, a myriad of algorithms is available. In this case, the original image (A) is segmented using (B) Munford & Shah [121] by its Megawave [60] implementation and (C) Gradient Network [152]. It is easy to see that little agreement between the two segmentation methods, regarding the number of regions and its delimitations.

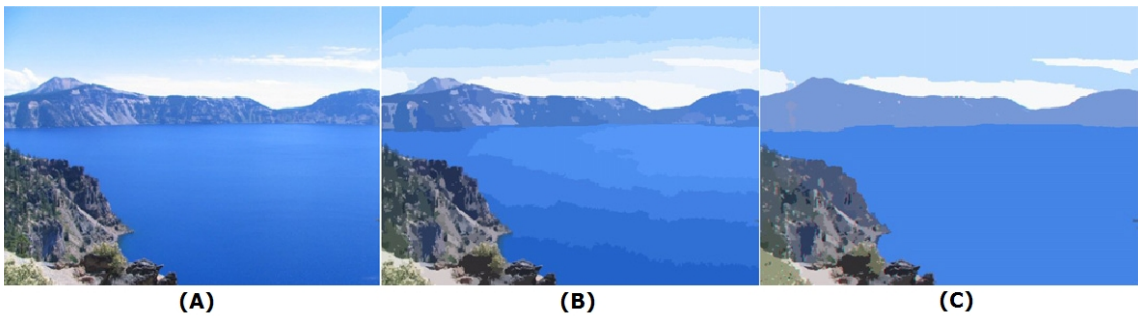


Figure 6.1. Image results for different segmentation algorithms

Many image segmentation methods also require parameters to be selected in order to achieve a good final segmentation result. Usually, supervised parameter learning is used to estimate a fixed parameter setting [151].

Recently, a new direction in image segmentation was taken. Instead of selecting one optimal parameter setting, it is proposed to combine several different segmentations, produced using different parameter settings, or different segmentation algorithms, into final consensus segmentation. This approach is known as image segmentation combination[1].

Initial efforts in image segmentation combination [3, 25, 51, 82, 87, 108] consider an image segmentation as a clustering of pixels. They apply standard ensemble clustering algorithms for segmentation combination. Each algorithm proposes the use of a different consensus function. More specifically, the methods proposed in [140] regarding graph-based ensemble clustering are used in [25, 87, 108] as consensus function. The differential factor between the referred methods is the ensemble generation step, in which K-Means with random cluster centroids, probabilistic sampling to generate fast segmentation and spectral clustering with random selected kernel values are the methods of choice, respectively. Jiang *et al.* [82] proposed the use of self-organizing maps (SOM) based segmentation [81] algorithm for ensemble generation. The segmentations in the ensemble are generated using a different SOM with parameters such as, learning rate and distance threshold assuming different values. However, the combination step is also done via graph-based consensus functions [140]. Another segmentation combination method [25] is given in the context of video shot detection method. It uses as consensus function the $CSPA$ algorithm [140]. In this method texture information is considered as another constraint on scale-invariant feature transformation. Voting schemes based consensus function, Fischer and Buhmann [51] uses bagging [22] technique with path-based clustering to address the robustness issue. This method requires a direct relabeling in order to achieve a consensus segmentation. All possible relabeling permutations ($K!$ in which K is the number of clusters) are produced and the one which maximizes the sum over the empirical cluster assignment probabilities from the previous mapping is selected as the new mapping configuration. This method presents a serious drawback since the computation of it for larger K values is non-practical. Finally, Aljahdali *et al.* [3] also proposed a method based on voting schemes. The main difference between it and the last approach is that, it uses different classifiers to generate the ensemble of segmentations.

Other methods specifically designed to deal with the image segmentation combination [174, 177] can be found in the literature. They take into account details such

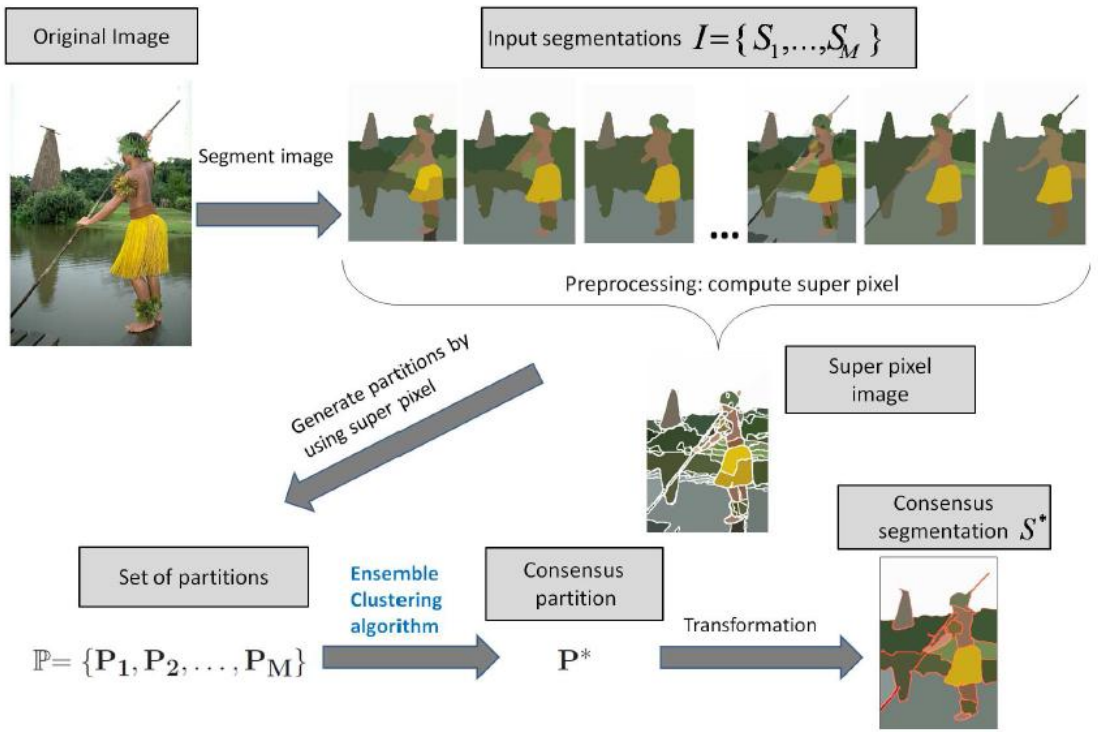
[1]In some papers, the terms image fusion and image merging are used. In this text the use the term image segmentation combination is preferred since the other terms can also appear in different contexts.

the size of the datasets (that can be a constraining factor for many combination approaches), and well as structured pattern's lattice. In these works, ensemble clustering methods are mostly used in combination with other heuristics. Quantitative experimental results are not provided or limited.

Wattuya *et al.* [162] proposed a new image segmentation combination approach. It uses a consensus function based on random walker to infer the final consensual partition. This work presents a remarkably detailed evaluation in which concepts as ensemble variability and accuracy are addressed in details. The biggest advantage of this method in relation the others discussed earlier is that it proposes an interesting way to address the problem of dataset dimensionality, the usage of multiple segmentation algorithms for the generation of the ensemble, and the proposition of an entirely new consensus function.

This chapter builds on the previously cited works and provides a broad experimental study in order to explore the capabilities of ensemble clustering methods applied to the context of image segmentation combination. As it was discussed before, different image segmentation combination methods present as general rule a different ensemble generation scheme, ranging from very simple as proposed in [25] to other very complex [162]. The final consensual image is usually achieved by means of a standard consensus function. The main contribution of the approach hereby proposed consists of applying and comparing a broad variety of widely used ensemble clustering methods to the image combination problem. Additionally, a comparison is presented with the supervised parameter learning approach. It shows that comparable or even superior results are received without knowing ground-truth. In order to make image datasets possible to be proceeded by such general clustering combination methods, some pre- and post-processing steps are required. This chapter proposes a way of doing so in a standard manner. The framework proposed allows the usage of virtually any consensus function to address the problem of image segmentation combination.

6.1 Framework for Image Segmentation Combination



Original Image
Input segmentations $I=\{S_1,\ldots,S_M\}$
Segment image
Preprocessing: compute super pixel
Super pixel image
Generate partitions by using super pixel
Consensus segmentation S^*
Set of partitions
Ensemble Clustering algorithm
Consensus partition
Transformation
$\mathbb{P}=\{\mathbf{P_1},\mathbf{P_2},\ldots,\mathbf{P_M}\}$
$\mathbf{P}^*$

Figure 6.2. Processing pipeline for image segmentation combination using ensemble clustering methods

In order to use any existing combination method to deal with image segmentation combination, the processing pipeline in Figure 6.2 is proposed.

- Produce M segmentations $I = \{S_1, \cdots, S_M\}$ of an image by varying parameters or using different segmentation algorithms;
- Generate super-pixels and eliminate small super-pixels to further reduce the number of patterns;
- Compute cluster ensemble $\mathbb{P}$ by using the super-pixels produced;
- Apply any general clustering combination method to $\mathbb{P}$ and receive a consensus clustering P^*;
- Post-processing step: P^* is transformed into a consensus segmentation S^*.

The M segmentations can be generated using any segmentation algorithm. In this thesis three segmentation algorithms are used. Different segmentations over the same image are achieved by varying the required parameters within a specified range. The remainder of this section reviews in detail the steps composing the framework above.

Pre-processing of the Image Segmentation Ensemble

Image datasets are known to contain a large number of objects (pixels). For instance, a common 640480 image contains 307200 pixels. For the purpose of image segmentation combination, this number is further enlarged by the number of the segmentation samples in the ensemble, leading to a considerable workload. Any useful combination method requires some sort of diminishment in the number of objects to be processed.

This framework proposes a pre-processing step motivated by the fact that neighboring pixels, which are equally labeled in all segmentations, do not have to be clustered individually by the ensemble clustering algorithm. It suffices to compute a representative object called super-pixel for each such group of pixels (image segment). This method was originally proposed in the context of image segmentation [136].

It is important to notice that the pixel-grid is not a natural representation of visual scenes. It is rather a representative imposition of the digital imaging process. A more natural and, possibly efficient way to represent images is to work with perceptually meaningful entities obtained from a low-level grouping process. For example, normalized cuts [129] can be supplied in order to partition an image into, N segments. This process is called super-pixel representation. It has many advantages:

- **Computationally efficiency**: By reducing the image complexity from hundreds of thousands of pixels to potentially only a few hundred super-pixels;
- **Representationally efficiency**: By using pairwise constraints between units, while only for adjacent pixels on the pixel-grid, can model much longer-range interactions between super-pixels;
- **Perceptually meaningfulness**: Each super-pixel is a perceptually consistent unit, i.e. all pixels in a super-pixel are most likely uniform in color and texture;
- **Near-completeness**: Super-pixels are results of an over-segmentation. Therefore, they tend to conserve most structures in the image. There is very little loss in moving from the pixel-grid to the super-pixel map.

The aim of super-pixel algorithms is to divide the pixels of the image into non-overlapping subsets of pixels (super-pixels) such that pixels in each super pixel are equally labeled. Figure 6.3 exemplify graphically the effect of computing the super-pixels of a synthetic image. Pictures (A), (B) and (C) represent three different image segmentations. Pictures (D), (E) and (F) the corresponding super-pixels. Colors are used to represent the produced super-pixels. Note that there are two large regions (white and green in the original image segmentation at the first row). Those regions are the same in all three segmentations. Consequently, they all are mapped to the same super-pixel (yellow and brown regions in row two). Intuitively, a new pixel can be represented into an existing super-pixel if and only if it shares the same label of the super-pixel among the different segmentations.

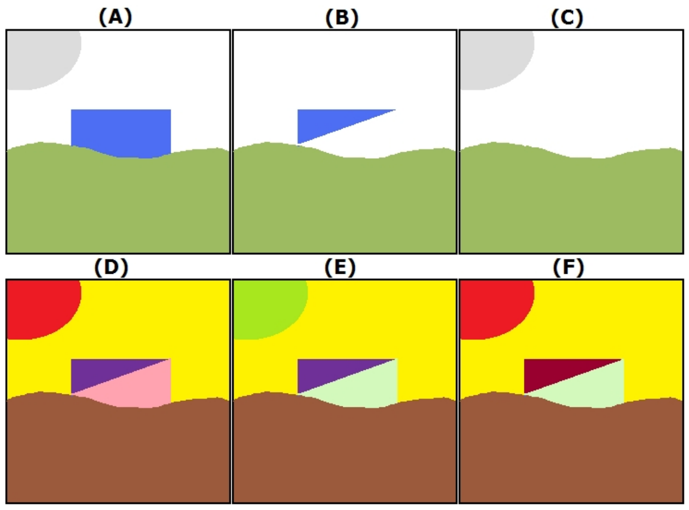


Figure 6.3. Example of super-pixel computation for synthetic image

Algorithm 6.2 has $O(MN)$ complexity for computing the super-pixels of an ensemble of segmentations. It receives as input a set I of image segmentations and returns a set Sp of super-pixels. There is one-to-one correspondence between elements of I and Sp. Initially, all image pixels are unassigned. The algorithm follows by picking a pixel at random. This pixel is used to create a new super-pixel, initially containing only this pixel. In the next step, the next unassigned pixel is selected as the reference pixel and it is assigned to the current super-pixel. Afterwards, the identification of the pixel's neighborhood takes place. Since images have a well behaved lattice, it is fairly quick to identify such neighborhood. 4- and 8-neighborhoods are viable options. The algorithm follows iterating over all neighboring pixels. A test is executed to verify if the reference pixel and the current neighbor pixel share the same label among all segmentations. If the answer is affirmative, the current neighbor pixel is assigned to the current super-pixel, otherwise, it continues by testing the

Algorithm 6.2 Algorithm for computation of the super-pixels

```
Input: an ensemble $I$ of image segmentations
Output: a set $Sp$ of super-pixels

1. Proceed until all pixels in $I$ are assigned to a super-pixel
2.     select the next pixel $p_u$ still unassigned to any super-pixel
3.     create a new super-pixel $sp_l$ containing initially only $p_u$
4.     While there are unassigned pixels in $sp_l$
5.         select next unassigned pixel $p_i \in sp_l$
6.         assign $p_i$ to the current super-pixel
7.         $\forall\ p_j \in$ neighborhood $\eta_p$ of $p_i$
8.             if $L(p_u) = L(p_j),\ \forall\ S_i \in I$
9.                 $sp_l \leftarrow sp_l \cup \{p_j\}$
10.            end
11.        end
12.    end
13. end
```

remaining neighbor pixels. Once all pixels in the current super-pixel are processed, another unassigned pixel is selected and a new super-pixel is created. The process continues until there are no more unassigned pixels.

Clustering Ensemble via Super-pixels

Once the set of super-pixels Sp is available, the set $\mathbb{P}$ can be easily computed by listing the labels of each super-pixel. A common representation of $\mathbb{P}$ is to create an ensemble matrix $CE_{N\times M}$ in which rows index individual patterns and columns individual partitions. The size of objects in $\mathbb{P}$ is at least the maximum number of segments in the original segmentations $S_i \in I$ and at most the number of pixels in the image, which is very unlikely.

Ensemble Combination

The next framework step refers to the actual application of consensus functions to combine $\mathbb{P}$ into a final consensual partition P^*. The experiments presented in this chapter, evaluated eleven consensus functions, namely BoK, $BOEM$, $WPCK$, $CSPA$, $HGPA$, $MCLA$, QMI, RW, SDP, EAC_AL and EAC_SL. These consensus functions are described in Chapter 3.

Post Processing

After applying a general clustering combination method to $\mathbb{P}$ a consensus clustering P^* is received. Using the same method to compute super-pixels, P^* is transformed into a consensus segmentation S^*. Because of the processing that eliminates small super pixels before computing $\mathbb{P}$, there are some unlabeled pixels. These pixels are merged to the neighboring region with the smallest color difference.

6.2 Experimental Results

This section describes the datasets used to evaluate the framework proposed. The experiments and evaluation measures are also detailed.

6.2.1 Datasets

The color images from the Berkeley dataset [112] are used by the experiments. This database is widely used for image segmentation evaluation and it is composed of 300 natural images of size 481×321. To evaluate each image in the dataset, 3 state-of-art image segmentation algorithms are used to generate 3 ensembles, $TBES$, UCM and $TBES\&UCM$ ensembles. Each ensemble is composed of 10 segmentations obtained by varying the parameter values of the segmentation algorithms used to generate the ensemble. $TBES$ ensembles are generated with the $TBES$ algorithm [137], which is based on the MDL-principle and has as parameter the quantization level (ε). This parameter is varied for the following values: $\varepsilon = 40, 70, 100, 130, \cdots, 310$ to obtain the 10 to obtain the 10 segmentations in the ensemble. Furthermore, UCM ensembles are generated with an image segmentation algorithms based on ultra-metric contour map (UCM) [4]. Its only parameter is the threshold l. The values of choice are $l = 0.03, 0.11, 0.19, 0.27, 0.35, 0.43, 0.50, 0.58, 0.66, 0.74$. Finally, $TBES\&UCM$ ensembles are generated by using two different segmentation algorithms: $TBES$ and UCM. Five segmentations are obtained with $TBES$ ($\varepsilon = 40, 100, 160, 220, 280$) and the others with UCM ($l = 0.03, 0.19, 0.35, 0.50, 0.66$).

6.2.2 Combination by Ensemble Clustering *vs* Supervised Learning

Considering the parameter selection problem in image segmentation, it is desirable to provide a general insight into the capability of general ensemble clustering methods. It is equally desirable to explore how well general consensus functions perform in the context of segmentation combination. For this reason, the process proceeds as follows.

Combination by ensemble clustering: the pre-processing step described in the last section is applied to each ensemble. Some ensemble clustering algorithms have a parameter K, which specifies the number of regions in the consensus result. This is the case for $CSPA$, $HGPA$, $MCLA$, EAC_SL, EAC_AL and SDP. Thus, for these algorithms for each ensemble K is set equal to the average number of regions of the images of the ensemble. The other algorithms BoK, $BOEM$, RW and $WPCK$ do not need any parameter specification. In the experiments, it was also used RW and $WPCK$ with a fixed K value (denoted by $RWfixed$ and $WPCKfixed$) for comparison purposes.

Supervised parameter learning: In order to gain further insight into the power of the framework we decided to apply supervised parameter learning to the same datasets. Therefore, to each dataset we compute the average performance measure over all 300 images of Berkeley dataset to each parameter setting. The parameter setting with the largest value is selected as the optimal fixed parameter setting for the corresponding dataset. By these means it is provided a quantitative comparison with the proposed approach.

6.2.3 Evaluation of Segmentations

In the experiments, the obtained results are compared to the human segmentations (ground-truth) of each image. Four well-known measures are used to evaluate the algorithm results: Normalized Mutual Information (NMI) [140], Variation of Information (VI) [114], Rand Index (RI) [128] and F-measure [112].

NMI, RI and F-measure are similarity measures that take values within the range $[0, 1]$, which 1 means a perfect correspondence between the segmentation and the ground-truth. On the other hand, VI is a dissimilarity measure that takes values in $[0, +\infty]$, and 0 means a perfect correspondence between segmentations. In order to show experimental results in a homogeneous way we present a dissimilarity version of the measures NMI, RI and F-measure. Therefore, the values $1 - SM$ are computed, which SM represents NMI, RI and F-measure respectively, whereas lower measure values mean better correspondence.

6.2.4 Ensemble Segmentation Results

Figure 6.4 shows some ensemble clustering results for free and fixed K. For the case in which K is fixed, the ensemble clustering algorithms EAC_AL, SDP and $WPCK$ perform similar (Figure 6.4 2 h - j) as it can also be seen by analyzing the performance values in Table 6.1. However, for free K the results may be very different (Figure 6.4 c - e) which is not surprising. In both cases the input segmentations are nicely combined. Based on the experiments presented, it is possible to conclude that, satisfying segmentation results may be achieved by using ensemble clustering methods (e.g. EAC_AL). The parameter selection problem can be solved to a certain degree. In this sense our benchmark pointed out some landmarks concerning the combination of segmentations and may be the base for future research.

The Berkeley database provides for every image several ground-truth segmentations. Because pairwise ground-truth segmentations for the same image can differ for the experiment, it is decided to handle this problem by evaluating the results using two different strategies in order to get objective results. First, the ground-truth image which yields the maximum performance value (denoted as "best GT") is reported. Secondly, the average (over all) performance values received from different ground-truths ("all GT") is presented.

Table 6.3 shows the ensemble clustering results for fixed parameter K.

Table 6.1 shows the results for algorithms with free parameter K. Ensemble clustering algorithms are applied to each dataset and performance of the consensus segmentation is evaluated by taking for each segmentation the ground-truth image which yields the maximum performance value ("best GT = best") and by averaging over all ground-truth images ("all GT = all"). Lower values are better. For NMI $WPCK$ outperforms the other ensemble clustering algorithms on all datasets and for VI, RW is the best for two datasets. For RI and F-measure $WPCK$ is best, whereas the less complex algorithm BoK only for VI yields very good results. Considering the results for fixed K in Table 6.3 it is observed that there is no considerable variability among NMI, RI and F-measure. If NMI, RI and F-measure are considered three algorithms outperform the others slightly: EAC_AL, SDP and $WPCK$. In contrast, for VI EAC_SL and $MCLA$ yield slightly better results. It is hard to judge why VI prefers these algorithms. Apart from its desirable properties the relevance of

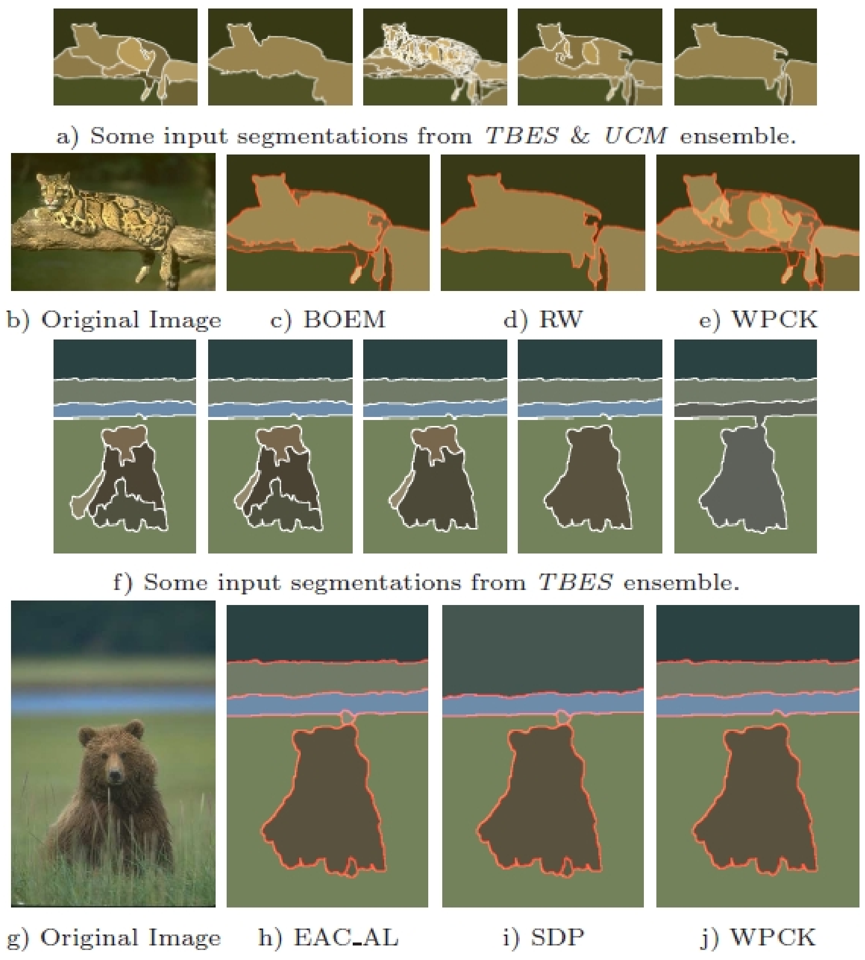

a) Some input segmentations from *TBES* & *UCM* ensemble.
b) Original Image
c) BOEM
d) RW
e) WPCK
f) Some input segmentations from *TBES* ensemble.
g) Original Image
h) EAC_AL
i) SDP
j) WPCK

Figure 6.4. Segmentation results for free K (c-e), and for fixed K (h-j)

VI for image segmentation is unclear and has to be further explored. For two methods (RW and $WPCK$) the results for fixed and free parameter K can be directly compared. In both cases the results for fixed K are better than the results for free K. However, it must be emphasized that in some situations heuristics for fixing K are insufficient and methods which adaptively select K are preferred. The results for supervised parameter learning are shown in Table 6.2. Considering the results for fixed K, for the $TBES$ and $TBES\&UCM$ dataset many ensemble clustering methods yield results close to those received by parameter learning. This is especially the case for EAC_AL, SDP and $WPCKfixed$. For NMI even better results are received for EAC_AL ($TBES\&UCM$ dataset). These results give raise to the assumption that good segmentation results may be received by using general ensemble clustering methods like EAC_AL, SDP or $WPCK$ without knowing ground-truth. In this context it must be emphasized that in many application scenarios supervised learning is not applicable because ground-truth is not available. Thus, ensemble

Table 6.1. Ensemble clustering results for free parameter K

		NMI		**VI**		**RI**		**F-meas.**	
Dataset	**Method**	best	all	best	all	best	all	best	all
FH ensembles	*BoK*	0.41	0.47	1.76	2.05	0.19	0.24	0.51	0.58
	BOEM	0.37	0.43	2.05	2.30	0.15	0.22	0.51	0.58
	RW	0.37	0.43	1.70	1.97	0.16	0.22	0.50	0.57
	WPCK	0.36	0.42	2.12	2.34	0.15	0.21	0.51	0.58
	Learning	0.35	0.41	1.58	1.87	0.15	0.21	0.47	0.54
TBES ensembles	*BoK*	0.41	0.48	1.34	1.73	0.21	0.28	0.66	0.63
	BOEM	0.35	0.42	1.52	1.82	0.16	0.22	0.45	0.52
	RW	0.49	0.55	1.57	1.97	0.28	0.34	0.58	0.64
	WPCK	0.32	0.29	1.58	1.85	0.15	0.22	0.42	0.49
	Learning	0.31	0.37	1.34	1.69	0.14	0.20	0.40	0.47
TBES & *UCM* ensembles	*BoK*	0.51	0.56	1.34	1.77	0.29	0.37	0.56	0.63
	BOEM	0.38	0.45	1.58	1.86	0.20	0.25	0.45	0.52
	RW	0.42	0.48	1.32	1.68	0.21	0.28	0.50	0.57
	WPCK	0.31	0.37	1.66	1.92	0.14	0.20	0.40	0.47
	Learning	0.29	0.36	1.29	1.62	0.13	0.18	0.23	0.41

clustering methods are preferred in scenarios where parameters of segmentation algorithms are unknown. To further illustrate the capability of the methods to each dataset the average ensemble performance AEP is determined which reflects the average quality of the image segmentation ensembles. The AEP is determined by computing the average performance value to each ensemble in a dataset and then, averaging over all these values (Table 6.2) in which lower values are better. Here we only note that e.g. for the $TBES\&UCM$ ensembles nearby all ensemble clustering algorithms yield better performance values than the average ensemble performance.

Table 6.2. Performance evaluation of supervised learning and average performance of ensembles

		NMI		**VI**		**RI**		**F-meas.**	
Dataset	**Method**	best	all	best	all	best	all	best	all
Supervised learning	*TBES*	0.31	0.37	1.34	1.69	0.14	0.20	0.40	0.47
	UCM	0.28	0.35	1.29	1.61	0.11	0.18	0.32	0.41
	TBES&UCM	0.29	0.36	1.29	1.62	0.13	0.19	0.33	0.42
Average ensemble performance	*TBES*	0.34	0.41	1.53	1.83	0.16	0.22	0.44	0.51
	UCM	0.36	0.42	1.88	2.25	0.17	0.24	0.42	0.51
	TBES&UCM	0.35	0.42	1.53	1.87	0.17	0.24	0.43	0.51

Table 6.3. Ensemble clustering results for fixed parameter K

		NMI		**VI**		**RI**		**F-meas.**	
Dataset	**Method**	best	all	best	all	best	all	best	all
FH ensembles	$CSPA$	0.36	0.42	2.28	2.51	0.15	0.32	0.52	0.59
	EAC_SL	0.45	0.51	1.79	2.08	0.24	0.29	0.51	0.58
	EAC_AL	0.36	0.42	1.94	2.17	0.15	0.21	0.51	0.58
	$HGPA$	0.40	0.46	2.63	2.85	0.18	0.25	0.55	0.63
	$MCLA$	0.36	0.42	1.88	2.13	0.14	0.21	0.50	0.57
	$RWfixed$	0.35	0.41	1.81	2.06	0.15	0.21	0.49	0.56
	SDP	0.36	0.42	2.22	2.45	0.15	0.22	0.51	0.59
	$WPCKfixed$	0.35	0.41	1.93	2.16	0.14	0.21	0.49	0.57
	Learning	0.35	0.41	1.58	1.87	0.15	0.21	0.47	0.54
$TBES$ ensembles	$CSPA$	0.33	0.39	1.75	1.99	0.14	0.21	0.42	0.49
	EAC_SL	0.33	0.39	1.43	1.71	0.16	0.21	0.42	0.49
	EAC_AL	0.32	0.39	1.51	1.78	0.15	0.21	0.41	0.48
	$HGPA$	0.32	0.38	1.75	1.98	0.14	0.21	0.42	0.49
	$MCLA$	0.34	0.41	1.47	1.77	0.16	0.22	0.44	0.51
	$RWfixed$	0.41	0.47	1.82	2.08	0.22	0.28	0.49	0.55
	SDP	0.32	0.38	1.91	2.16	0.14	0.21	0.41	0.48
	$WPCKfixed$	0.32	0.39	1.53	1.80	0.15	0.20	0.41	0.48
	Learning	0.31	0.37	1.34	1.69	0.14	0.20	0.40	0.47
$TBES$ & UCM ensembles	$CSPA$	0.32	0.38	2.14	2.42	0.14	0.22	0.61	0.48
	EAC_SL	0.29	0.36	1.46	1.74	0.13	0.19	0.35	0.43
	EAC_AL	0.28	0.35	1.59	1.86	0.12	0.19	0.35	0.43
	$HGPA$	0.34	0.40	2.27	2.56	0.15	0.22	0.43	0.51
	$MCLA$	0.34	0.40	1.41	1.71	0.17	0.22	0.57	0.49
	$RWfixed$	0.30	0.36	1.69	1.97	0.13	0.20	0.37	0.45
	SDP	0.29	0.36	1.72	2.00	0.13	0.19	0.37	0.45
	$WPCKfixed$	0.30	0.36	1.66	1.93	0.13	0.20	0.38	0.45
	Learning	0.29	0.36	1.29	1.62	0.13	0.19	0.33	0.42

Part II

A New Consensus Function for Ensemble Clustering

Chapter 7

Sum of Pairwise Distances

Finding the median representative of a set of objects is a task of great importance. Its value is mainly given by the fact that a median of a set is usually a good way to represent the data as a single instance. To compute the median of numbers one can simply rely on the widely known statistical tools. However, to compute the median of complex object sets such as graphs [48, 72, 78], strings [103], images [1, 162] and volumes [2] more sophisticated methods are required.

A widely used method to address the problem of finding the median of objects is given by the generalized median formulation presented in Chapter 2. It was shown that the median value possible to be obtained by the generalized median is computationally intractable for many reasonable distance functions. Consequently, any method based on this formulation uses an heuristic to reach a suboptimal solution.

By investigating different ways to reach the set's representative with the hope that better minima could be obtained, *the sum of pairwise distances - SoPD* was born. In this context, the idea of minimizing the distance between pairs of objects is considered. By initially finding a pairing of objects and minimizing specifically such dissimilarities it is expected to reach a more robust heuristic to solve the problem of finding a median object, especially in cases where the objects variability is accentuated.

This chapter introduces a new formulation for the problem of finding the median of objects. Section 1 discusses the motivational idea behind the *SoPD*. Afterwards, the problem is mathematically formulated. The implementation aspects of *SoPD* are the subject of Section 2. The steps of initial object selection, weights computation, computation of most dissimilar pairs or objects and the optimization of the cost function are presented. Section 3 presents the requirements to apply *SoPD* in the context of ensemble clustering. Two consensus functions are presented and the implementation aspects properly described. Section 4 proposes a cluster validity index based on $SoPD$. In Section 5, an experimental comparison between $SoPD$ and other well known consensus functions is presented. This chapter ends in Section 6 with a discussion and final remarks

7.1 Definition

The diagram presented in Figure 7.1 helps to motivate the $SoPD$ idea. It depicts a set comprised of eight points (blue dots), a representative point obtained by the generalized median (green dot) and the true median value (red dot) in a given vector space $\mathcal{P}$.

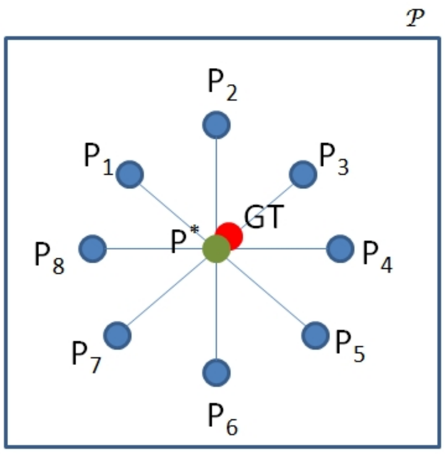


Figure 7.1. Optimal example to compute the median partition

Consider the example given in Figure 7.1. From a geometrical point of view in an Euclidian space, it is easily seen that in this optimal situation, the median object P^* is located in the exact intersection point between the opposite points $\{(P_1, P_5), (P_2, P_6), (P_3, P_7), (P_4, P_8)\}$.

Unfortunately, one cannot expect to find such perfect conditions in dealing with real problems. Situations such as the one presented in Figure 7.2 are more likely to occur. In cases such as this, the set of objects simply do not have enough information to allow an extrapolation close to the optimal solution based solely on the identification of the intersection point.

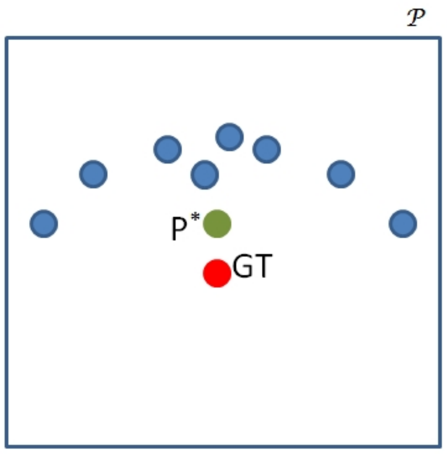


Figure 7.2. More realistic example of real object's distributions found in real-life problems

The idea hereby introduced proposes to find the pairs of objects that are most likely to be the opposite of each other (most dissimilar). Subsequently, a cost function based on such pairs is minimized.

$SoPD$ is based on the idea that opposite or most dissimilar objects should be located opposite to each other. Therefore the mid-distance between the objects would be most likely the location of the median result. Of course in real applications these conditions are rarely met. Therefore, provided the list of dissimilar pairs is known, this condition could be relaxed to meet the triangle inequality.

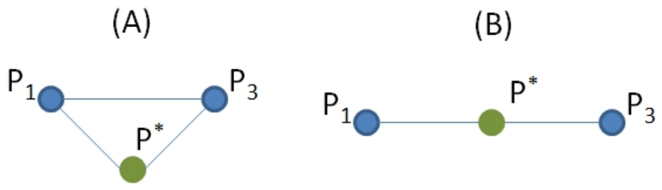


Figure 7.3. Triangle inequality applied to the pairwise distance between partitions

Figure 7.3 shows how the midpoint situation could be relaxed to satisfy the triangle inequality. In the optimal case (B) the sum of the two smaller sides of the triangle would be equal to the size of the larger side $(d(P_1, P_3) = d(P_1, P^*) + d(P^*, P_2))$. However, in the real case (A), a new measure of distance needs to be defined in order to properly assess this new condition.

The sum of pairwise distances method can be formally defined as follows. Let $\mathbb{P} = \{P_1, P_2, \ldots, P_M\}$ be a set of M objects. The set of all possible objects is denoted by $\mathcal{P}_X$ $(\mathbb{P} \subset \mathcal{P}_X)$. $d(p, q)$ is a distance function defined between any given objects $\xi, \varrho \in \mathcal{P}_X$. Additionally, it is required of $d(.,.)$ to be a metric for reasons that will become clear later. The input set $\mathbb{P}$ can be partitioned into $M/2$ pairs. And there are $\frac{M!}{2^{\frac{M}{2}}\cdot(\frac{M}{2})!}$ possible such partitions. Let Φ be the set of all such partitions. The elements $\varphi \in \Phi$ are given by:

$$\varphi \;=\; \{(\xi_1, \varrho_2), (\xi_3, \varrho_4), \cdots, (\xi_{M-1}, \varrho_M)\} \tag{7.1}$$

The set φ has $\phi = \lfloor|\mathbb{P}|/2\rfloor$ pairs and $\lfloor\cdot\rfloor$ is the floor function representing the integer part of the number. The number of elements in $\mathbb{P}$ is required to be even, since the method requires $\mathbb{P}$ to be partitioned into pairs. Cases where the number of elements M in $\mathbb{P}$ is odd can be handled similarly to the solution proposed in [78].

The goal of the the sum of pairwise distance method is to find an object $P^* \in \mathcal{P}_X$, which optimally represents the set of objects $\mathbb{P}$.

$$(P^*, \varphi^*) = \arg \min_{\tilde{P}\in\mathcal{P}_X, \varphi\in\Phi} \sum_{(\xi,\varrho)\in\varphi} \left| d(\xi, \varrho) - d(\xi, \tilde{P}) - d(\tilde{P}, \varrho) \right| \tag{7.2}$$

Equation (7.2) presents a very complex problem since there are two variables to be optimized, namely, the median object and φ^* (set of pairs of objects). The complexity is straightened by the fact that the search space for these two variables is considerably large. Expectation maximization (EM) algorithm is suitable for solving this problem.

The next section presents a detailed definition of the *SoPD* by addressing in details all of its steps.

7.2 Computational Method

The computation of *SoPD* requires various steps. The most intuitive implementation of it uses the traditional EM algorithm. This is the model followed by the optimization algorithms presented in this chapter. This section starts with schematics for the *SoPD* model and follows by addressing all its computational steps.

Figure 7.4 depicts the general steps for the *SoPD* method. The set of objects ($\mathbb{P}$) is used to infer an initial median value $\tilde{P}$. Both $\tilde{P}$ and $\mathbb{P}$ are then used to find the pairings of most dissimilar objects. Afterwards, the cost function *SoPD* is minimized producing a new candidate solution P'. It is evaluated regarding its fitness as final solution. If it is deemed acceptable, it will be promoted as the approximated solution P^* and the procedure stops returning P' as result. However, if the stop criterion is not satisfied, the process will be retrofitted and a new pairing of most dissimilar objects will be produced starting the process all over again.

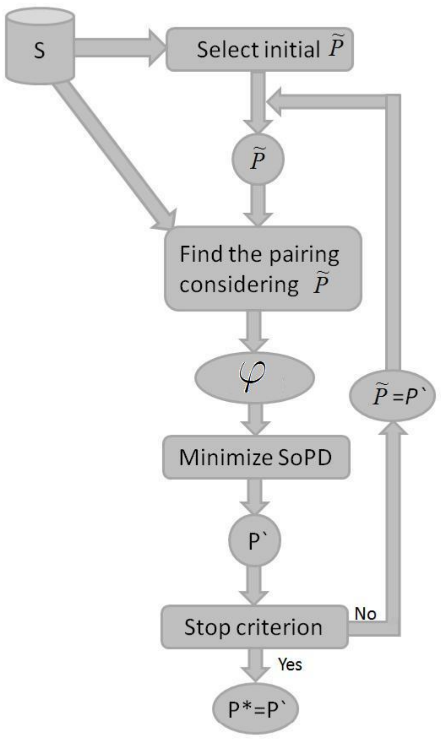


Figure 7.4. Fluxogram for the computation of $SoPD$

The following subsections discuss systematically the aspect of each step of the $SoPD$ method.

7.2.1 Initial Candidate Solution

Similarly to the generalized median, the sum of pairwise distances cost function originally given by Equation (7.2) is a computationally intense problem, requiring a suboptimal solution to be considered. Most optimization methods work under the assumption an initial solution is known. This initial solution is used as a starting point for the optimization method. $SoPD$ is no different. It requires an initial partition $\tilde{P}$ in order to compute the set φ of most dissimilar pairs of objects.

In the experiments performed later on this chapter, it is observed that an initial good candidate partition will have a positive impact in the overall process. Additionally, it induces a quicker convergence compared to the situation in which a bad initial candidate is provided. A good candidate partition is expected to be one located around the middle of the ensemble distribution (see Subsection 3.2.2 to review the idea of ensemble variability). Figure 7.5 presents the results given in error rate for 10 different initial candidate partitions over 8 UCI-Irvine datasets. The data are organized by initial partitions (x-axis) and error rate obtained (y-axis). For this experiment, the ensemble generation scheme KMclicksrandK40_07 is used. The initial partitions are selected at random among the 40 available partitions in the ensemble.

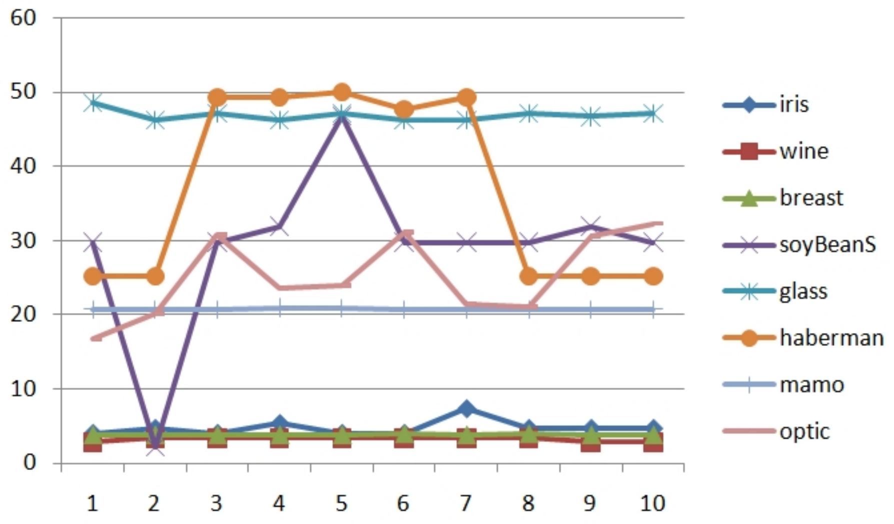


Figure 7.5. Different initial candidate partitions

As it can be seen, for most datasets there are little impact in selecting any of the partitions. The best attainable result will be invariably reached. For the haberman and soybean datasets, however, this is not the case. A considerable degradation in the final results can be observed for some initial partitions.

In the experiments presented later on this chapter, the initial partition is set to be the result of the simple *BoK* consensus function (or the set median of the ensemble).

7.2.2 Weight Computation

In order to compute the set of most dissimilar pairs of objects φ, a measure of dissimilarity between objects is required. This is done by computing the pairwise weight between all possible pairs of objects in $\mathbb{P}$. This step is required in order to provide enough background information for the maximum graph matching algorithm described in the next subsection.

The weighting function is derived from the cost function proposed, and defined as follows:

$$w_{\xi,\varrho} = \left| d(\xi, \varrho) - d(\xi, \tilde{P}) - d(\tilde{P}, \varrho) \right| \tag{7.3}$$

where $w_{\xi,\varrho}$ stands for the weight between the objects $\xi, \varrho \in \mathbb{P}$.

A matrix M_w of size $M \times M$ is computed using Equation (7.3) for all possible pairs of objects in $\mathbb{P}$. M_w is the input needed by the next step of the *SoPD* method. The distance relationships between all possible pairs of objects can be viewed as an undirected connected graph. As it can be seen by inspecting Equation (7.3), the weight between two objects is set to be the absolute value of the distance between ξ and ϱ minus the distances of ξ and ϱ to $\tilde{P}$.

The weight between two objects informs how different they are from each other in relation to the given candidate object. It is important to notice that this has no direct correspondence to the examples given in the introductory part of this chapter. It is also heavily dependent on the distance function $d(\cdot, \cdot)$ used. The weighted Equation (7.3) is envisioned to work with the Mirkin and VI metric distances (see Chapter 2). It is also possible to use the Euclidean distance. For that, the objects need to be remapped to an Euclidean space. This is possible via MDS as reviewed in Chapter 3. However, a different weight function needs to be established.

7.2.3 Finding the Pairs

Before describing the method for finding the most dissimilar pairs of objects an explanation of what is considered a "most dissimilar pair" is in order. The objective is to find a binary relationship between objects in the set where the sum of all their pairwise distances/weights is maximum. Additionally, it is also desirable that the objects in each pair to be as distant/dissimilar to each other as possible.

This can be achieved by computing the *maximum graph matching*, provided an appropriate edge's weight is available representing the relationship between vertices. Quickly reviewing, the graph matching is defined as follows. Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a matching Mt in $\mathcal{G}$ is a set of pairwise non-adjacent edges. This means that no two edges share a common vertex. A vertex is matched (or saturated) if it is incident to an edge in the matching. Otherwise the vertex is unmatched.

A maximal matching is a matching Mt of a graph $\mathcal{G}$ with the property that if any edge not in Mt is added to Mt, it is no longer a matching, that is, Mt is maximal if it is not a proper subset of any other matching in graph $\mathcal{G}$. In other words, a matching Mt of a graph $\mathcal{G}$ is maximal if every edge in $\mathcal{G}$ has a non-empty intersection with at least one edge in Mt.

However, in this case the problem faced is a weighted graph. A maximum weighted matching is defined as a perfect matching where the sum of the values of the edges in the matching has a maximum value.

The maximum weighted matching algorithm requires as input a weighted graph. The matrix M_w introduced in the last section is the intended input. By the end of the process, the algorithm outputs a matching Mt that can be directly regarded as the pairing φ of most dissimilar objects of $\mathbb{P}$.

7.2.4 Optimization Techniques

During a traditional expectation step, in which $\tilde{P}$ is known (the initial guess), the set φ is computed as presented in the last subsection. The maximization step computes the $SoPD$ as given in Equation (7.4).

$$SoPD(\tilde{P},\varphi) = \sum_{(\xi,\varrho)\in\varphi} \left| d(\xi,\varrho) - d(\xi,\tilde{P}) - d(\tilde{P},\varrho) \right| \tag{7.4}$$

Theoretically, any optimization technique available could be used to minimize the cost function proposed above. However, it must considered that after each iteration there is the possibility that a new set of pairs of patterns needs to be recalculated, in order to match the new $\tilde{P}$. This can impose a considerable workload in the optimization procedure. For this reason, it is proposed to update the set of pairs only after a given distance Δ between the $SoPD$ value of the initial candidate partition and the current one be surpassed. In such case, instead of always working with the exact set of pairs matching $\tilde{P}$, for a number of iterations, the minimization function will be using a slightly imprecise set of pairs. This is however corrected once the pairing of objects is updated. A test is made in order to decide if the set of pairs should be updated based on Δ and a threshold parameter. Nevertheless, the gain in computational complexity is considerable.

This thesis implements two optimization techniques using the $SoPD$ definition.

Best one Element Moves: this method is based on the optimization technique proposed to solve ensemble clustering problems [67] (see Chapter 3 for more details about this method).

Simulated Annealing: Introduced in the mid of 1970s by Scott Kirkpatrick *et al.* [89] it was originally developed to better optimize the design of integrated circuit (IC) chips by simulating the actual process of annealing. It was inspired by the process of annealing in metallurgy. This technique involves heating an alloy and systematically cooling it in a controlled way in order to reduce its defects. Similarly, the optimization method considers at each step a randomly selected small neighborhood perturbation and probabilistically decides if moving the system to the perturbed state would minimize the cost function. The probabilities are selected in a way that the system will tend to move to states of lower energy.

The next section presents the cited optimization techniques applied to the context of ensemble clustering.

7.3 *SoPD* Applied to Ensemble Clustering

For the purpose of presenting the *SoPD* applied to ensemble clustering two optimization methods are selected, namely *BOEM* and simulated annealing. The code for the *BOEM* function is given by Algorithm 7.3.

Algorithm 7.3 *BOEM* based *SoPD* ensemble clustering algorithm

```
Input: the ensemble $\mathbb{P}$ of partitions to be combined,
       the initial candidate partition $\tilde{P}$
Output: a consensus partition $P^*$

01. Repeat while minimum not found
02.     Compute $M_w$ given $\tilde{P}$ and $\mathbb{P}$
03.     Find the set $\varphi$ via maximum graph matching given $M_w$
04.     bestSoPD $\leftarrow$ $SoPD(\tilde{P}, \varphi)$
05.     $\forall\ p \in \mathbb{P}$
06.         check if the assignment of $p$ to any cluster other than $C(p)$ results
            in $SoPD(\tilde{P}, \varphi) <$ bestSoPD
07.         If YES
08.            keep the new cluster assignment for $p$
09.            bestSoPD $\leftarrow$ $SoPD(\tilde{P}, \varphi)$
10.         Else
11.            try the next possible cluster assignment for $p$
11.     End
12. End
13. $P^* \leftarrow \tilde{P}$
```

Line 1 ensures that the procedure will continue until the function is deemed as optimized. The stop criteria can be maximum number of iterations, no improvement in the minimization function, etc. In Line 2 the matrix of weights M_w is computed as described in Subsection 7.2.2, given as input the clustering ensemble, an initial candidate partition, and the distance function between partitions. Line 3 is responsible for finding the pairing φ of most dissimilar partitions as described in Subsection 7.2.3. The current $SoPD$ value is computed in Line 4, using the initial candidate partition and the pairing of partition. This is the baseline $SoPD$ value that will be minimized. Line 5 iterates over all patterns of the current partition. A test is performed (Line 6) to check if the $SoPD$ value is achieved by assigning the current pattern to a cluster other than the on it was originally assigned ($C(p)$ gives the cluster of p). If the change produces a better $SoPD$ value, the pattern is definitively changed to the new cluster and value of bestSoPD is updated (Lines 7-9). If the change in cluster assignment fails to improve the $SoPD$, the algorithm follows by trying the next possible cluster assignment for p (Lines 10 and 11). The algorithm proceeds until all possible assignments for all possible clusters are tried. Once the algorithm reaches this point, a test for convergence is made, indicating if it should continue or stop returning P^* as consensus partition (Line 13).

The second consensus function uses the simulated annealing method. The fluxogram for the $SoPD_{sa}$ algorithm can be seen in Figure 7.6. There are two major processes that take place originally belonging to the simulated annealing method and two additional, introduced by the $SoPD$ formulation. At the beginning, the pairing based on $\tilde{P}$ and the corresponding $SoPD$ is computed. This value is used to assess if the progressively proposed candidate partitions P' achieve better results than $\tilde{P}$. The general simulated annealing method states that for each temperature, a number of cycles will take place. This number can be defined as a parameter or specified by the algorithm's designer. As a cycle runs, the inputs are randomized. Only randomization steps which produce a better set of inputs are retained. In the case of ensemble clustering, the randomization can be achieved by changing the label of a random pattern within a given label's range.

The process responsible for finding the initial pairing of most dissimilar partitions takes place as described in Subsection 7.2.3. It receives as input the ensemble of partitions ($\mathbb{P}$) and an initial candidate partition $\tilde{P}$. The output of this process is a pairing φ of most dissimilar partitions. The process continues by computing the $SoPD$ according to the Equation (7.4). Following the fluxogram, a test is performed to assess if the new candidate partition is more accurate than the current one. This test is based on the $SoPD$ value returned by Equation (7.4). For the case in which the value is smaller than the one computed for $\tilde{P}$, the partition P' is attributed to $\tilde{P}$. After the specified number of training cycles is completed, the temperature is lowered. The algorithm continues by determining whether or not the temperature

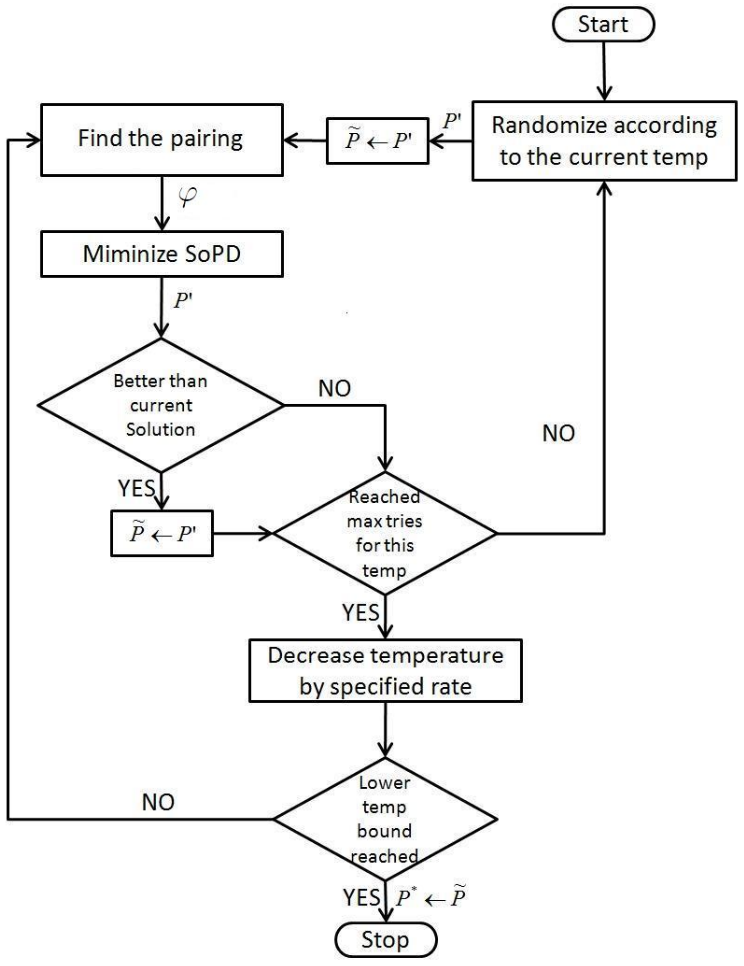

Start
Randomize according to the current temp
P'
$\tilde{P} \leftarrow P'$
Find the pairing
φ
Miminize SoPD
P'
Better than current Solution
NO
YES
$\tilde{P} \leftarrow P'$
Reached max tries for this temp
NO
YES
Decrease temperature by specified rate
Lower temp bound reached
NO
YES
$P^* \leftarrow \tilde{P}$
Stop

Figure 7.6. Fluxogram for the simulated annealing based $SoPD$ consensus function

has reached the lowest temperature allowed. If the temperature is not lower than the lowest temperature allowed, the temperature is decreased and another cycle will take place. At this point, a new set of most dissimilar pairs of partitions is computed. If the temperature is lower than the lowest temperature allowed, the simulated annealing algorithm terminates by attributing $P^* \leftarrow \tilde{P}$ as consensus partition.

$BOEM$ and $SoPD_{sa}$ display similar performance for all tests executed. The real difference observed is regarding the computational time required. For experimental purposes only the results of $SoPD_{sa}$ are presented.

7.4 *SoPD* Validity Index

Chapter 2 and later on Chapter 3 present a number of cluster validity indexes. Some of them operate over the partition and a given ground-truth and others are specifically designed to evaluate ensemble clustering methods [44, 140]. Similarly to the last category, it is also possible to define a CVI based on the formulation proposed in this chapter. It is named sum of pairwise distances validity index ($SoPDvi$). It receives as parameter the consensus partition P^* to be evaluated and the pairs of patterns (ξ, ϱ) computed as described in Subsection 7.2.3 using as reference partition P^*. The Equation (7.5) shows how to compute the $SoPDvi$.

$$SoPD_{vi}(\varphi, P^*) = \frac{\sum_{(\xi,\varrho)\in\varphi} |d(\xi,\varrho) - d(\xi, P^*) - d(P^*,\varrho)|}{M/2} \tag{7.5}$$

The objective of $SoPD$ is to infer a consensus partition P^* presenting minimum distance to the set of pairs of partitions. Therefore, smaller values of $SoPDvi$ represent better solutions degrading progressively as the value increases. The range of the values itself will be heavily dependent on the distance function used.

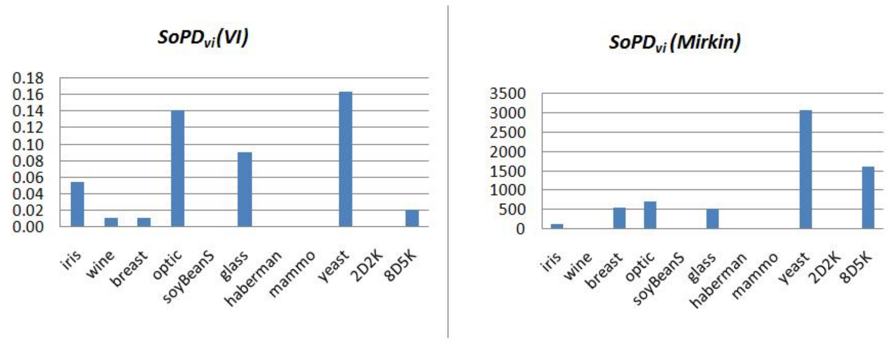


Figure 7.7. $SoPDvi$ computed using VI and Mirkin distances

Figure 7.7 shows the experimental values of $SoPDvi$ computed for 11 UCI-Irvine datasets. Two similarity measures are used, VI and Mirkin. The data are organized datasets (x-axis) and $SoPDvi$ values (y-axis). For this experiment, the ensemble generation scheme KMclicksrandK40_07 is reported. Small values represent little variability. As it can be seen, for some datasets such as soyBeanS, haberman, mammoMass, and 2D2K the values are really small what can be directly translated as poor variability in the ensemble. By inspecting the $SoPDvi$ values for other ensemble generation methods it was attested that different datasets will achieve higher variability by means of different generative methods. This is the main reason why the results regarding ensemble clustering methods presented in this thesis report the average of the 40 ensemble generation methods described in Chapter 2.

7.5 Experimental Results

In order to properly evaluate the results obtained with the $SoPD_{sa}$ consensus function a direct comparison against the random walker consensus function in both versions with known and unknown number of target clusters is presented. The results of other well known consensus functions such as evidence accumulation, both using single-link and average-link and the graph based methods $HGPA$ and $CSPA$ are also reported. The results presented in Tables 7.1, 7.2, 7.3, and 7.4 are the average values of 40 different ensembles (see Chapter 2 for a review of the ensemble generation schemes) each one computed 10 times. The number of partitions in each ensemble varies from 10 to 40. Results for percentage of errors and for the VI index are presented for each dataset evaluated. The initial partition provided to the $SoPD_{sa}$ algorithm is the set median of the ensemble.

Regarding the toy datasets (Tables 7.1 and 7.2) it is possible to see that $SoPD_{sa}$ scores as good as or slightly better than any other consensus function considered except the celipsiod dataset for which the VI index of $RWfix$ is better than $SoPDsa$. However, to the error rate they both missed exactly the same (1.9%).

Table 7.1. VI index for the toy datasets

Dataset	EAC_SL	EAC_AL	$HGPA$	$CSPA$	RW	$RWfix$	$SoPD_{sa}$
8D5K	0.00	0.00	4.55	0.00	0.02	0.02	0.00
2D2K	0.27	0.27	2.00	0.32	0.27	0.27	0.27
celipsoid	1.68	1.68	1.99	1.67	1.69	1.52	1.62
twoRings	1.97	1.97	1.99	1.99	1.97	1.97	1.84
scattered	1.84	1.84	1.86	1.86	1.84	1.84	1.84
halfrings	1.14	1.14	1.97	1.29	1.14	1.14	1.14

Table 7.2. Error rates (in %) for the toy datasets

Dataset	EAC_SL	EAC_AL	$HGPA$	$CSPA$	RW	$RWfix$	$SoPD_{sa}$
8D5K	0.00	0.00	77.10	0.00	0.10	0.10	0.00
2D2K	1.90	1.90	49.40	2.30	1.90	1.90	1.90
celipsoid	27.56	27.56	49.78	27.11	28.00	26.02	27.56
twoRings	47.51	47.51	48.62	48.90	47.51	47.51	46.11
scattered	43.18	43.18	48.48	46.97	43.18	43.18	43.18
halfrings	13.75	13.75	49.81	17.84	13.75	13.75	13.75

Tables 7.3 and 7.4 present the results for the UCI-Irvine datasets in which the overall performance of $SoPD_{sa}$ is better. For those datasets, some considerable

Table 7.3. VI index for the UCI-Irvine datasets

Dataset	EAC_SL	EAC_AL	$HGPA$	$CSPA$	RW	$RWfix$	$SoPD_{sa}$
balance	2.63	2.52	2.77	2.70	1.66	2.52	1.63
breast	0.95	0.46	1.93	1.13	0.40	0.40	0.40
control	1.36	1.48	2.41	1.31	1.30	1.60	1.23
ecoli	1.82	2.23	3.28	2.91	2.10	2.36	2.26
glass	1.90	1.99	3.08	3.38	1.33	1.27	1.31
haberman	1.28	1.28	1.83	1.83	0.83	0.91	0.87
heart	1.86	1.86	1.99	1.91	1.86	1.86	1.86
ionosphere	1.73	1.73	1.90	1.77	1.74	1.74	1.65
iris	0.73	0.49	2.15	0.54	0.13	0.17	0.13
lung	1.42	1.42	1.55	1.54	1.41	1.44	1.37
mammo	1.42	1.42	2.00	1.51	1.44	1.44	1.42
optic	2.09	1.70	3.73	1.66	0.77	0.56	0.53
parkinsons	1.25	1.25	1.80	1.74	1.25	1.25	1.25
post-op	1.96	1.98	2.45	2.47	1.72	1.94	1.89
protein	1.56	1.57	3.24	2.89	1.64	1.43	1.54
segmentation	1.82	1.82	2.34	1.74	1.97	2.05	1.82
sonar	1.94	1.95	1.99	1.97	1.96	1.94	1.87
soyBeanS	1.28	1.28	1.28	1.67	0.51	1.07	0.75
spect	1.46	1.55	1.73	1.57	1.55	1.54	1.44
spectf	1.24	1.24	1.73	1.60	1.23	1.23	1.21
taeval	2.91	2.91	3.04	2.97	2.87	2.87	2.87
tic-tac-toe	1.47	1.87	1.93	1.91	1.47	1.87	1.47
transfusion	1.38	1.40	1.79	1.76	1.40	1.40	1.40
wine	0.34	0.34	1.58	0.56	0.19	0.19	0.19
yeast	2.53	3.53	5.29	4.63	1.88	1.45	1.42

improvement can be observed in approximately 50% of the datasets, for 40% of the datasets the results are comparable to the best cases among the other consensus functions and for a few cases, other consensus functions scored better such as in the case of ecoli dataset where both VI and error ratio are considerably better for the EAC_SL consensus function. The same happens to the segmentation dataset, in which $CSPA$ scored better, gain for both VI and error rate.

The cases in which $SoPD_{sa}$ scored worst or similar to other consensus functions are investigated in details. It is possible to identify by inspecting each of the 40 ensembles that little variability in such ensembles exists. The variability assessment is made two-fold: a) computing the C_{var} index for each ensemble and by plotting the

Table 7.4. Error rates (in %) for the UCI-Irvine datasets

Dataset	EAC_SL	EAC_AL	$HGPA$	$CSPA$	RW	$RWfix$	$SoPD_{sa}$
balance	33.28	39.68	57.28	46.08	29.60	39.68	27.48
breast	34.85	3.81	49.63	17.28	3.22	3.22	3.22
control	39.34	37.70	44.16	21.71	47.40	38.76	22.24
ecoli	34.17	44.60	64.78	58.12	40.34	46.58	43.33
glass	48.13	48.60	58.88	64.49	44.86	42.52	43.73
haberman	25.16	25.16	49.67	48.04	26.47	22.88	25.16
heart	38.28	38.28	47.65	39.16	38.30	38.30	38.27
ionosphere	30.68	30.68	42.81	33.95	31.11	31.11	28.41
iris	32.00	4.67	35.00	5.33	4.00	4.67	4.00
lung	26.94	26.30	30.46	29.54	26.76	27.13	22.63
mammo	20.72	20.72	48.19	21.69	21.08	21.08	20.72
optic	54.00	27.20	57.40	17.50	21.50	11.90	11.72
parkinsons	27.88	27.88	47.76	41.44	27.76	27.76	27.13
post-op	43.53	43.76	61.38	63.28	40.26	43.33	42.70
protein	50.69	50.75	56.40	51.81	55.00	48.06	50.69
segmentation	43.81	43.81	39.05	28.57	44.29	42.38	43.78
sonar	43.98	44.10	48.34	44.17	44.59	44.33	42.31
soyBeanS	29.79	29.79	29.79	34.04	19.15	27.66	21.28
spect	39.63	43.38	49.42	37.55	43.95	44.09	38.94
spectf	36.13	35.98	49.41	39.40	35.51	35.51	35.17
taeval	52.98	52.98	60.26	50.33	53.64	53.64	53.11
tic-tac-toe	39.62	46.54	50.00	45.59	39.60	46.54	40.76
transfusion	30.87	31.09	48.36	45.01	31.03	31.01	31.10
wine	2.81	2.81	47.75	6.18	1.69	1.69	1.62
yeast	68.40	53.30	79.18	72.37	42.25	34.64	32.47

ensembles using the technique proposed in Subsection 3.2.2. This indication that little variability in the ensemble has a negative impact in the usage of the $SoPD_{sa}$ motivated the inspection of cases such as iris or yeast datasets in which $SoPD_{sa}$ is clearly superior. It is observed that such ensembles present a high C_{var} value and the partitions in the ensemble when plotted using the MDS technique are well distributed around a central point. These facts lead to the conclusion that $SoPD_{sa}$ is a good choice for cases where a good variability in the ensemble exists.

7.6 Discussion

This chapter presented a new method to compute the median of sets of objects. The goal of the the sum of pairwise distance method is to find an object which optimally represents the set of objects provided as input.

For implementation purposes, $SoPD$ requires an initial candidate median value $\tilde{P}$. The method follows by computing the weights between all possible pairs of objects. Different distance functions will require consequently different weighting functions. Once the weights are computed, the set of most dissimilar pairs of objects is computed. The method requires that the input set must have an even number of objects. In cases in which the original number of objects is odd, various actions can be taken. A suitable method refers to the identification of the pair of most similar objects and the discard of one of them. Once the pairing of most dissimilar objects is found, the method can be optimized by virtually any optimization function. The experimental results presented shows the usefulness of $SoPD$.

$SoPD$ was mainly presented within the context of ensemble clustering. This is due to the fact that ensemble clustering is the main subject of this thesis. However, it is believed that $SoPD$ is a valid approach to address other applications in which the average value needs to be computed. A possible field of application is the computation of median graphs [78].

The median graph problem is formulated as follows. Given a set S of graphs $\mathcal{G}_i = \{\mathcal{V}, \mathcal{E}, \mu, v\}$ in which $\mathcal{V}$ is the set of vertices, $\mathcal{E}$ the set of edges, v and μ mapping functions binding labels l from a given finite alphabet L for nodes and edges. The median graph can be addressed via the median partition formulation (see Section 2.5). The only required change is that a suitable distance function $d(\cdot, \cdot)$ between graphs must be provided. Possible distances are the maximum common subgraph - MCS [94] and graph edit distance [6]. Feasible implementations of it can be given by $BOEM$ or set median. More advanced approaches such as genetic algorithms are given e.g. in [48] which also presents ways of efficient computation of median graphs based on the median partition formulation. $SoPD$ can be easily used to replace the standard SoD commonly used in this case.

Another promising development is to operate in continuous space, namely by means of graph embedding into vector spaces. In the literature, there is a number of embedding approaches such as in [104, 130] which proposed graph embedding using spectral features extracted from the graph's adjacency matrix. In [166] the embedding of graphs using Reimannian manifolds is investigated (see [48] for more details about graph embedding).

All those methods have in common the fact that any type of graph embedding will require three steps:

1. Embedding to a vector space;
2. Median vector computation;
3. Re-mapping the median vector to the median graph in the graph space.

Regarding the usage of $SoPD$ with vector space embedding, there are no restrictions in substituting the optimization function by one based on $SoPD$.

Sum of pairwise distances has proven to be a valid approach in addressing the problem of ensemble clustering. It also seems to be a viable option in the context of median graph computation, opening a wide range of applications. The possibility of embedding graphs into vector spaces allows access to mathematical properties not available in graph spaces. Additionally, the median value computation is greatly simplified in vector spaces. However, there is the additional cost of embedding the data to vector space and re-mapping it afterwards to the graph space, justifying its application in cases where the computation of then median can be too costly. Additionally, the steps of weights computation and identification of most dissimilar pairs of objects can be greatly simplified in vector spaces.

Part III

Constrained Ensemble Clustering

Chapter 8

Constrained Clustering

The number of works related to unsupervised constrained clustering has grown during the last few years. Recently, Basu *et al.* [15] compiled a book about the subject. However, the area still lacks of a comprehensive taxonomy. Constrained clustering is an extension of general clustering. It accommodates side information translated as constraints aiming to improve the clustering process. It takes advantage of known information about the data set to aid the clustering process. Partially labeled data, expected maximum and minimum cluster size, and pairwise linkage pattern relationship are examples of information used by a constrained clustering algorithm.

Constrained clustering method can be classified by two possible criteria:

a) Based on the types of constraints used;

b) Based on the way the constraints are used by constrained clustering algorithms.

In order to simplify the understanding of the reviewed methods presented later, this paragraph formally states the clustering problem. The task of clustering is to assign a label $L(p_i) \in 1, \cdots, K$ to each pattern (i.e. data point) p_i of a given dataset S of N patterns. It assumes a known number K of possible labels and a metrical space (S, d). Patterns with the same label L_j form the cluster C_j, which is represented by its cluster center μ_j. Generally, the concept of similarity is realized by a distance function d on the data space.

Most clustering algorithms operate with only very limited knowledge about the data they are supposed to cluster. The minimum information available required refers to a distance/similarity function $d(\cdot, \cdot)$ between objects in S, and some inner structure or algorithmic structure that is able to partition the data into K classes.

Additional information possibly available regarding the problem to be solved cannot be used by such algorithms since they are simply not designed to handle such information. Constrained clustering allows the usage of this extra information. Constraints can be inferred from side knowledge by a number of ways. Taking as an example the pairwise instance level constraints, Any such binary relation known about the dataset can be encoded as pairwise constraints. Another example of direct inference of constraints can be found in the field of constrained clustering applied to image segmentation, as presented by Luo *et al.* [107].

The next section presents a description of each type of constraint as well as the methods to incorporate them into clustering algorithms. It follows by addressing the topic of constraints relevance and closes with a presentation of the concept of transitive closure.

8.1 Types of Constraints

It is hard to define a comprehensive taxonomy of existing constrained clustering methods. However, some common features can be identified, allowing the creation of a systematic classification to help to organize the existing methods. There is a variety of side information passive to be translated as constraints. Furthermore, many standard clustering algorithms were already adapted in order to work in the presence of constraints. In this section, a description of the existing types of constraints is presented.

Instance Level Constraints

In a natural way, the terms 'similarity' and 'dissimilarity' lead to two symmetrical, binary relations $\sim$ (similar) and $\nsim$ (dissimilar). Furthermore, the relation $\sim$ should be reflexive and transitive. Both characteristics are not valid for the relation $\nsim$. A pattern is obviously not dissimilar to itself and the transitivity of $\nsim$ is logically not justified. The similarity (resp. dissimilarity) relation to the patterns p and q is represented by a must-link (resp. cannot-link) constraint $p \sim q$ (resp. $p \nsim q$). A must-link (resp. cannot-link) constraint postulates the same (resp. different) labeling for both patterns, that means:

$p \sim q \Rightarrow L(p) = L(q)$ and $p \nsim q \Rightarrow L(p) \neq L(q)$

Constraints are represented as sets of unsorted pairs $\mathcal{ML} := \{(p,q)|p \sim q\}$ and $\mathcal{CL} := (p,q)|p \nsim q$. User knowledge should extend the distance-based definition of similarity. Therefore must-links are valuable in situations in which the patterns are similar but located far away (see Figure 8.1). On the other hand cannot-links are useful to separate patterns belonging to different clusters but located close to each other.

Cardinality Constraints

It is possible to use constraints to control the maximum and minimum number of patterns in each cluster [17] easing the problem of creation of empty clusters and clusters with just a few patterns. This usually occurs when the dimensionality or the number of clusters is large.

Topological Constraints

Topological constraints dictate that the minimum/maximum distance between patterns must not be violated in order to allow them to be collocated into the same cluster. In [34] the constrained cop-KM is further generalized to accommodate two new kinds of constraints, ε and δ. The ε-constraint enforces that each pair of patterns can be classified into the same cluster if the distance between them is at most ε. Similarly, the δ constraint enforces that two patterns found in different clusters must be separated by a distance at least δ. These constraints can be seen as a generalization of must-links and cannot-links. An interesting part of this work refers to the algorithms provided to check the feasibility of the clustering problem in the presence of different types of constraints.

Soft Constraints

It is also possible to soften the instance level constraints in order to represent "preferences" instead of pairwise relation certainty. This variant is called soft constrains. In this new case, hard constraints no longer apply, since they are simply a small subset of soft constraints. A constraint can be represented by a triple $< p_1, p_2, s >$ in which the first two parameters are the constrained patterns and s is the strength of the constraint. It can be set to have different range of values, although the most usual is $[-1, 1]$. A must-link constraint $p_1 \sim p_2$ is equivalent to $< p_1, p_2, 1 >$. Similarly, cannot-link constraints such as $p_1 \nsim p_2$ is equivalent to $< p_1, p_2, -1 >$. In [155], a soft version of K-Means is presented. It argues a simple test of constraint violation as used by the hard constrained version of its algorithm no longer applies. Instead, a weighted constrained violation function is proposed, in which the strength of the violated constraints are taken into consideration. Finally, the objective function is changed to take into consideration the intra-cluster variance and the new constraint violation function. Soft constraints are also used within a mixture model framework [96]. A new EM algorithm is proposed, tuned specifically to deal with the additional soft constraint information. They also present some experimental results showing that it can be superior in the presence of noise compared to methods based on hard constraint. Finally, Leone *et al.* [97] extends the AP - Affinity Propagation algorithm to accommodate soft constraints. The method addresses the difficulty hard constraints have to be used with the AP algorithm due to its strongly reliance on cluster shape regularity. The method is tested over gene-expression data. There are no guarantees about non-violation of constraints when a final clustering is achieved.

Fully Constrained Dataset

This type of constraints is very similar to pairwise constraints. A binary relationship indication if pairwise patterns are most likely to be in the same cluster or in different clusters. It is usually represented by a complete graph that the edge's labels can have two possible values: a) $+$ indicating the patterns are similar, and b) $-$ if the patterns are dissimilar. Algorithms using this kind of constrained datasets operate strictly in maximizing the number of agreements or complementary, minimizing the number of disagreements. Its accuracy is usually measured by the number of constraints' violations. Most of the works using this type of constraint such as in [8, 39, 65] require a similarity function over the dataset in order to infer the binary relationships. The function outputs only two values namely, similar or dissimilar and it is used to create a complete graph of relationships of the dataset.

8.1.1 Ground Truth Based Constraints Generation

Most of the works proposing constrained clustering algorithms rely on synthetically generated constraints, in order to evaluate its performance.

Basu *et al.* [14] have presented an active learning algorithm for exploring the dataset S by a farthest-first traversal. The idea is to generate good user queries to define constraints. This method was used to define a quality measure for constraints in this work. Roughly speaking, the automatic constraint generation takes as input the ground-truth and the original dataset. Constraints are generated following two basic rules: (a) must-links with higher distances, and (b) cannot-links with lower distances. The complete exploration of the whole dataset, in order to find the farthest and closed distances is, however, computationally intensive. In order to solve this problem, the algorithm explores only a portion of the dataset selected at random. The automatic constraint generation algorithm must be fed with three parameters. $L, N_{\mathcal{ML}}$ and $N_{\mathcal{CL}}$. The first parameter (L) instructs the algorithm to pick ($L\%$) of the data at random, the distances are computed and finally the $N_{\mathcal{ML}}$ pair with higher distance and same label are picked as must-links. Similarly, the

Algorithm 8.4 Must-link constraints generation algorithm

```
Input: $X$ : Dataset
       $N_{\mathcal{ML}}$: Number of desired must-link constraints
       $P_{GT}$: a ground-truth associated to $X$
Output: The set $\mathcal{ML}$ containing $N_{\mathcal{ML}}$ must-link constraints

1. compute the pairwise distance $D_{X \times X}$ between all patterns in $X$
2. sort $D_{X \times X}$ in ascending order
3. repeat until $|\mathcal{ML}| < N_{\mathcal{ML}}$
4.   $(a, b) \leftarrow$ pair of patterns with maximum pairwise distance in $D_{X \times X}$
5.   if the pair of patterns (a,b) have the same label in the ground-truth
6.     $\mathcal{ML} \leftarrow \mathcal{ML} \bigcup \{(a, b)\}$
7.   end
8.   remove $(a, b)$ from $D_{X \times X}$
9. end
```

$N_{\mathcal{CL}}$ with smallest distances are picked as cannot-links. This is the method later used by this thesis in generating constraints for the constrained ensemble clustering methods proposed in Chapter 9.

Algorithm 8.4 shows how to compute the set of must-link constraints. An initial effort in computing the pairwise distance between all possible pairs of patterns needs to be made. In line two the distances are sorted in ascending order to facilitate the next steps of the algorithm comprising the identification of the pairs of patterns farther apart. The algorithm continues by trying to identify possible must-links by inspecting if the selected pair of patterns shares the same label in the ground-truth. The process continues until the desired number of must-links is obtained.

The set of cannot-links can be obtained in a similar manner, simply changing the search for maximum distance to minimum, and the test regarding the sharing of same labels to different labels. Additionally, the constraints generation program can be optimized by computing the sets of must and cannot-link constraints together.

Note that the automatic generation of constraints based on known ground-truth is a valid tool commonly used in many constrained clustering papers in order to evaluate the proposed algorithms correctness. For a discussion about used defined constraints, it will be referred to [155] and to Chapter 11 of this thesis, where constraints are user defined to drive the fiber segmentation process.

8.1.2 Constraints Relevance

Several works in the field of constrained clustering show that constraints can improve considerably the results of a variety of clustering algorithms. However it was equally shown that there is a large variation in this improvement, when using the same number of constraints for the same dataset. Wagstaff *et al.* [37, 158] have investigated this phenomenon and introduced two indexes (informativeness and coherence) in the attempt to provide further insight.

More intuitively speaking, it is easy to understand the problem of constraints relevance by restricting the discussion to pairwise linkage constraints. Consider the example given in Figure 8.1.

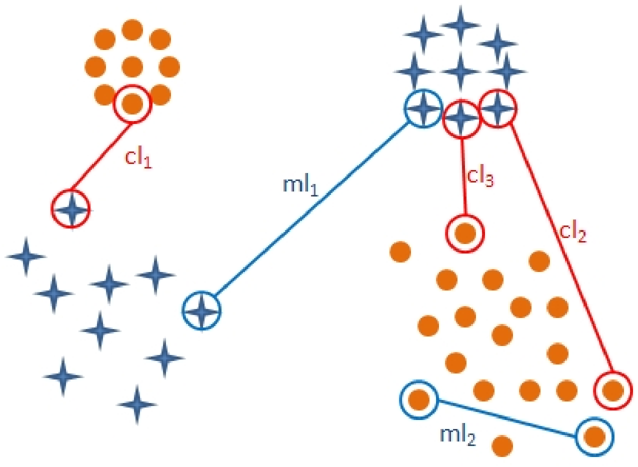


Figure 8.1. Examples of relevant and irrelevant constraints

Must-link constraints are relevant if they are able to relate patterns that would not be clustered together by an unconstrained algorithm. Similarly, cannot-link constraints would have its usefulness maximized, if they constraint patterns that would be collocated into the same cluster but that should not be. Simply putting, useful must-links are expected to occur between farther patterns, and cannot-links closer patterns. Figure 8.1 shows both examples of relevant and irrelevant constraints. The constraints ml_1, cl_1 and cl_3 are examples of useful constraints. The constraint ml_1 constraint patterns belonging to two distinctly separated sub-clusters. Similarly, cl_1 is defined between two patterns with a high probability to be picked as bellowing to the same cluster. The usefulness of ml_2 and cl_2 is minimized, since there is a high chance of the information encoded by those constraints to be irrelevant.

Ideally, a constrained clustering algorithm should be able to perform better compared to its unconstrained version. Moreover, it is also expected that the overall performance of the algorithm should improve with the increasing of the number of constraints. Both assumptions unfortunately are not necessarily correct. In fact, for cases that the constraints are considered locally such as in Wagstaff *et al.* [156] such violations do occur. However, global methods such as Rothaus *et al.* [132] tend to require less constraints and still show performance improvement with the increasing number of constraints. A quantitative comparison of some constraining methods is presented in the last section of this chapter.

8.1.3 Transitive Closure

It is very important to account for constraints consistency in generating constraints or when a set of constraints is provided prior to its usage by any constrained clustering algorithm. In cases where contradictory constraints exists, there is a good chance of most algorithms will end up in deadlocks, or unable to satisfy the whole set of constraints, leading to a failed partition. Therefore, there is the need to carefully check if any constraint is inconsistent to the whole set and to check if new constraints can be inferred from the existing ones. In Figure 8.2, an example of constraint inconsistency is presented. Black dots represent objects; blue lines represent must-link constraints and the red line, cannot-link constraints. There is no valid way to accommodate those three constraints since by allocating P_1, P_2 and P_3 into the same cluster the cannot-link constraint between P_2 and P_3 will be forcefully violated. A similar situation will arise by any attempt of fulfilling two giving constraints.

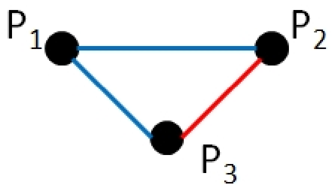


Figure 8.2. Example of invalid constraint

However, the verification of constraints consistency such as the inference of new constraints based on the existing, it can be achieved by the computation of the transitive closure over the sets of constraints $\mathcal{ML}$ and $\mathcal{CL}$. The purpose of such computation is to ensure the triangle inequality property is not violated.

In graph theory, transitive closure can be thought as constructing a data structure that makes it possible to answer reachability questions. It can be better understood by the examples given in Figure 8.3. Consider the following two must-link constraints: $P_1 \sim P_2$ and $P_1 \nsim P_3$. Consider now that no constraint must be violated. This fact implies objects P_2 and P_3 must be collocated into the same cluster as well, consequently a new constraint is inferred, $P_2 \sim P_3$. Similarly, the $P_1 \sim P_2$ and $P_1 \nsim P_3$ constraints are considered. It is easy to see that P_2 and P_3 cannot be put into the same cluster otherwise the $P_1 \nsim P_3$ constraint would be violated. Finally, in a case where the constraints $P_1 \nsim P_2$ and $P_1 \nsim P_3$ are given there is no possible inference about the relationship between P_2 and P_3.

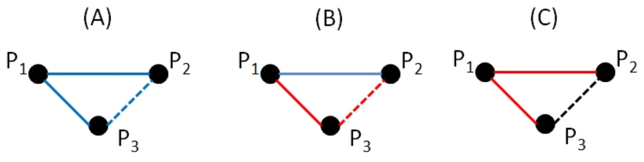


Figure 8.3. Inference of new constraints

The computation of the transitive closure is very important to speed up the clustering process as well to ensure no deadlocks occurs, given an invalid classification done in an early stage of the clustering process. Table 8.1 summarizes the rules for computation of the transitive closure.

Table 8.1. Rules for computing the hard transitive closure

constraint 1	constraint 2	produce
$p_1 \sim p_2$	$p_2 \sim p_3$	$p_1 \sim p_3$
$p_1 \sim p_2$	$p_2 \nsim p_3$	$p_1 \nsim p_3$
$p_1 \nsim p_2$	$p_2 \sim p_3$	$p_1 \nsim p_3$
$p_1 \nsim p_2$	$p_2 \nsim p_3$	–

The rules applied to infer new constraints based on the existing ones can also be used to check if the constraints in the set are inconsistent. It is easy to come up with an algorithm that checks the four rules given in Table 8.1. By finding an inconsistent constraint, , it can be chosen simply to remove it, making the set of constraints once again consistent.

The transitive closure computation can also be extended to deal with soft constraints. Table 8.2 show the extended rules.

In this case, either the maximum or minimum values can be used. However, by realizing those binary relationships, it seems to make more sense to use the minimum values.

Table 8.2. Rules for computing the soft transitive closure

constraint 1	constraint 2	produce
$< p_1, p_2, s_1 >$	$< p_1, p_3, s_2 >$	$< p_1, p_3, min(s_1, s_2) >$
$< p_1, p_2, -s_1 >$	$< p_1, p_3, s_2 >$	$< p_1, p_3, -min(s_1, s_2) >$
$< p_1, p_2, s_1 >$	$< p_1, p_3, -s_2 >$	$< p_1, p_3, -min(s_1, s_2) >$
$< p_1, p_2, -s_1 >$	$< p_1, p_3, -s_2 >$	$-$

8.2 Constraining Methods

Constrained clustering algorithms can be also evaluated based on its mechanics or inner algorithms structure. Figure 8.4 organizes the methods in three main branches, namely a) invasive; b) data space transformation; and c) distance transformation methods. In this section, a short review is presented of some methods related to the three categories.

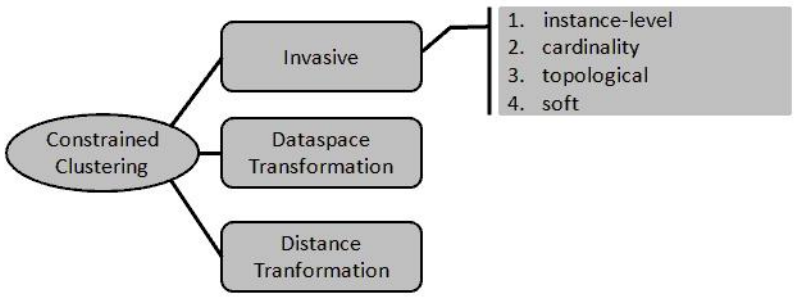


Figure 8.4. Comprehensive taxonomy of constrained clustering methods

Invasive

Invasive methods explicitly adjust clustering algorithms so that the available side information can be properly incorporated into the clustering framework. The algorithms must be adapted in such a way that the problem specific constraints take precedence over the algorithms' build-in objective function. Before any label assignment is made, a validation check is performed in order to access if any existing constraint is violated. Invasive methods usually work under the assumption that the set of constraints provided is regarded as correct classification information over the data. Therefore, they need to be fully satisfied.

Figure 8.5 shows the general architecture of invasive constrained clustering. The dataset and the set of constraints are provided as input. The constrained clustering algorithm is usually a version of standard clustering algorithm adapted to validate constraints before the label assignment step. The algorithm outputs a partition of the dataset, in cases that the full set of constraints can be fulfilled or an empty set $\{\}$ otherwise.

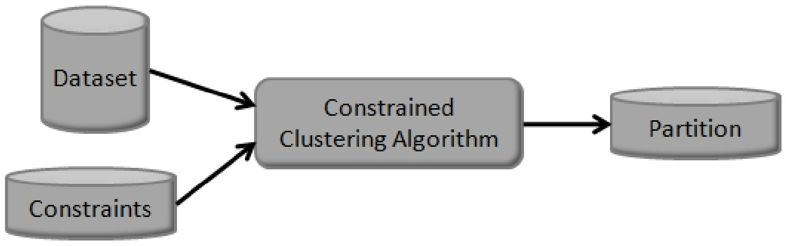


Figure 8.5. Constrained clustering architecture

Examples of such extensions are given by the COP-COBWEB method [156], and [157] which extends the K-Means algorithm by adding a check during the assignment step to test if constraints are violated. Another example is the constrained version of hierarchical clustering [35, 36, 38]. Similarly to [157], a constraint violation test is executed before merge two clusters. Another example of invasive method is the PC-KM [14]. This is yet an extension of the KM algorithm that focusing on the idea that a good initialization can greatly improves the clustering performance as shown in [13]. The algorithm starts by computing the transitive closure over the $\mathcal{ML}$ and $\mathcal{ML}$ sets. Neighborhood sets are created and for to each set of connected components in $\mathcal{ML}$. An heuristic is proposed to adapt the number of neighborhood sets to the desired K number of target clusters. PC-KM is essentially an EM algorithm. It alternates between the pattern assignment step and the computation of the cluster centers. The process continues until a convergence criterion is reached. In the patterns assignment step, a constraint violation value is computed. The assignment is made based on the minimal number of constraints violated.

Data-Space Transformation

Data space transformations are characterized by a known transformation $T \rightarrow X : \tilde{X}$. In the new space $\tilde{X}$, the constrained clustering can be done with a simple distance function $\tilde{d}$ (see [90]). Constrained spectral clustering (CSC) perform a data space transformation. Giving a similarity measure between patterns a weight matrix is computed. The general idea is to embed pairwise constraints in the spectral formulation, usually within the weight's matrix in order to build up a new adjusted data space that encompasses the given constraints. More information about spectral clustering can be found in [84]. Examples of constrained spectral clustering methods are given in [168, 169, 178].

Distance Transformation

This type of constraining method relies on the idea of modify a distance measure d to accommodate a set of instance-level constraints (namely must-links and cannot-links) in a way that the desired properties of the similarity measure are preserved (see [167]). Methods based on this type of constraining strategy rely heavily on unsupervised algorithms which are able to learn implicitly metrics that take the input dataset and find an embedding of it in some space. Examples of such algorithms are the Locally Linear Embedding (LLE) [133] and multidimensional scaling (MDS) [30].

Examples of constrained clustering using distance transformation can be found in [27] that KL divergence, adapted using gradient descent, and Mahalanobis distances, trained using convex optimization [9, 167] are applied. Those metric-based semi-supervised clustering algorithms exclude unlabeled data from the metric training step, as well as separate metric learning from the clustering process. Also, existing metric-based methods use a single distance metric for all clusters, forcing them to have similar shapes. MPCK-Means [19] incorporates both metric learning and the use of pairwise constraints. It performs distance-metric training utilizing both unlabeled data and pairwise constraints. Finally, Rothaus *et al.* [132] is yet another distance transformation based constrained clustering method. It proposes an algorithm which is able to converge to an improved clustering result in the presence of just few constraints. This is achieved by means of spreading the influence of pairwise constraints throughout the neighborhood of constrained patterns, allowing logical sub-clusters to be merged in more general groups. The method uses a new distance transformation that interprets constraints as ”shortcuts” between sub-clusters.

Correlation Clustering

Correlation clustering uses a complete constrained graph encompassing all patterns in the dataset. The edges are labeled $+$ (to patterns that must be placed in the same cluster) and $-$ (to patterns that cannot be placed into the same cluster). No information about distances is available, but only the similarity or dissimilarity between patterns. The focus is to find clusters that maximize the number of agreements as well the complementary minimize the number of disagreements. Examples of methods based on this formulation can be found in [8, 39, 65].

8.3 Experiments

In order to further evaluate the impact of considering constraints locally and globally as discussed earlier a quantitative comparison of, COP-KM [157], DMLM - Distance Metric Learning Method proposed by Xing *et al.* [167] and COPGB-KM [132] is presented. COP-KM is a method used extensively as comparison based on many publications about this topic. It considers constraints locally and ensures no violations. The second and third methods spread the influence of constraints globally.

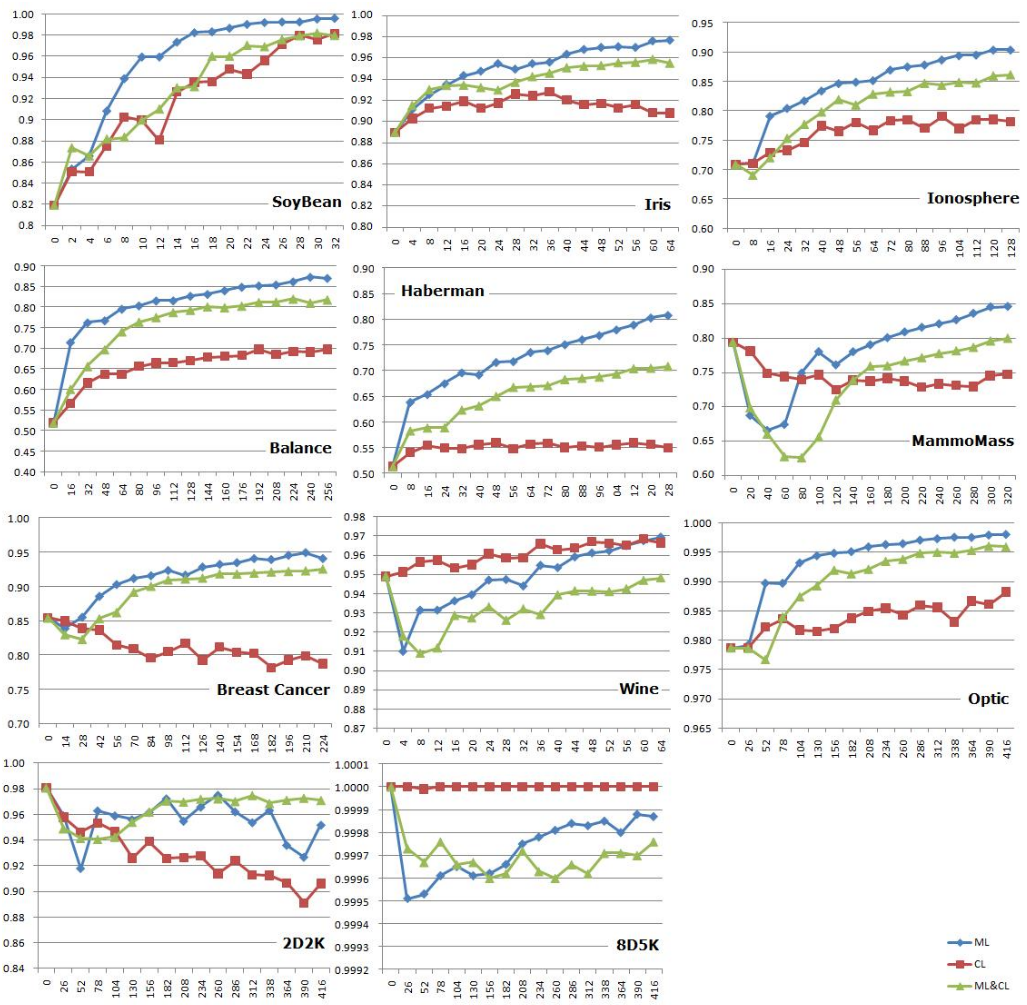


Figure 8.6. Results of the COPGB-KM for the eleven test datasets

To each dataset three different experiments are performed: (a) considering only must-links; (b) considering only cannot-links; and (c) considering both types of constraints. To (a) and (b) only the results of COPGB-KM and COP-KM are available, since DMLM requires both types of constraints in order to work properly.

Sixteen test constraint settings are generated to each of the three experiments, totalizing forty eight constraint settings. The number of must-links $N_{\mathcal{ML}}$ and cannot-links $N_{\mathcal{CL}}$ is computed based on the total number of patterns to each dataset. The result is averaged for 100 test runs per setting. For each dataset, COPGB-KM and COP-KM are clustered 4900 times (an additional test is made to encompass the case that there are no constraints) and DMLM 1600 times. In total, the experiments created 136800 partitions.

The parameter L described in the last section is set to search randomly 40% of the total number of patterns of each dataset.

Five different quality measures (accuracy [167], NMI, precision, recall, and F-measure [14]) are computed to compare the clustering results with the ground-truth.

Since the five measures show similar behavior to all datasets, only the accuracy is reported giving a direct measure of the goodness of the classification in respect to the ground-truth[1].

The first envisioned experiment evaluates how the COPGB-KM algorithm behaves in the presence of different number and kinds of constraints. Figures 8.7 and 8.7 shows a comparison of how the algorithm performed in the presence of only must-links, only cannot-links and in the case that both types of constraints are present. Diamond shaped markers represent results considering only must-link constraints; square shaped the results of only cannot-links; and triangles the results using both types of constraints. Axis X represents the number of constraints used and Y axis the accuracy. The scale is adjusted individually to each dataset. The number of constraints is computed based on the total number of patterns of each dataset. COPGB-KM performs better in the presence of only must-link constraints (diamond shape markers) compared to both other cases. For the case that both types of constraints are present, an average quality between the best case (only must-links) and the worst case (only cannot-links) is observed.

Exceptions to this general observation are the MammoMass and the 2D2K datasets. In those cases, for a small number of constraints, the case that only cannot-links are present performs slightly better if compared to the only cannot-link case.

The Wine dataset is a complete exception. Only cannot-links perform almost always better than only must-links and in the situation which both types of constraints are present shows a worst result.

[1]Error rate indicates the percentage of patterns wrongly classified. Accuracy gives the percentage of patterns classified correctly.

Another interesting remark refers to the general behavior of the accuracy curve for all possible combination of constraint types. The accuracy tends to increase with the number of constraints used. However, we observe that the introduction of fewer constraints induce an accentuated increase in accuracy followed by a less considerable increase afterwards. This is an important fact, since it can be interpreted as a minimal requirement of side information.

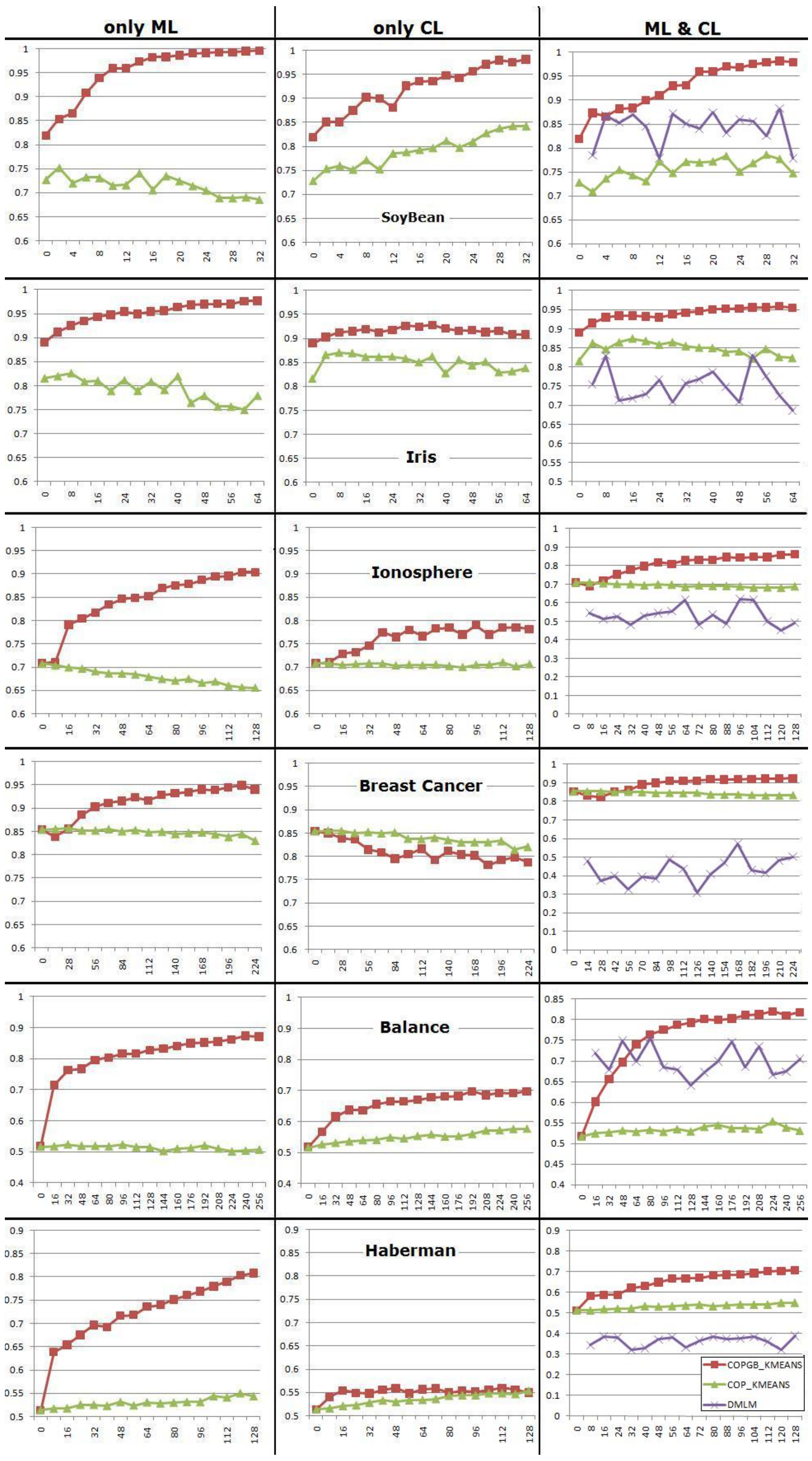


Figure 8.7. Comparison of COP-KM, COPGB-KM and DMLM constrained clustering methods

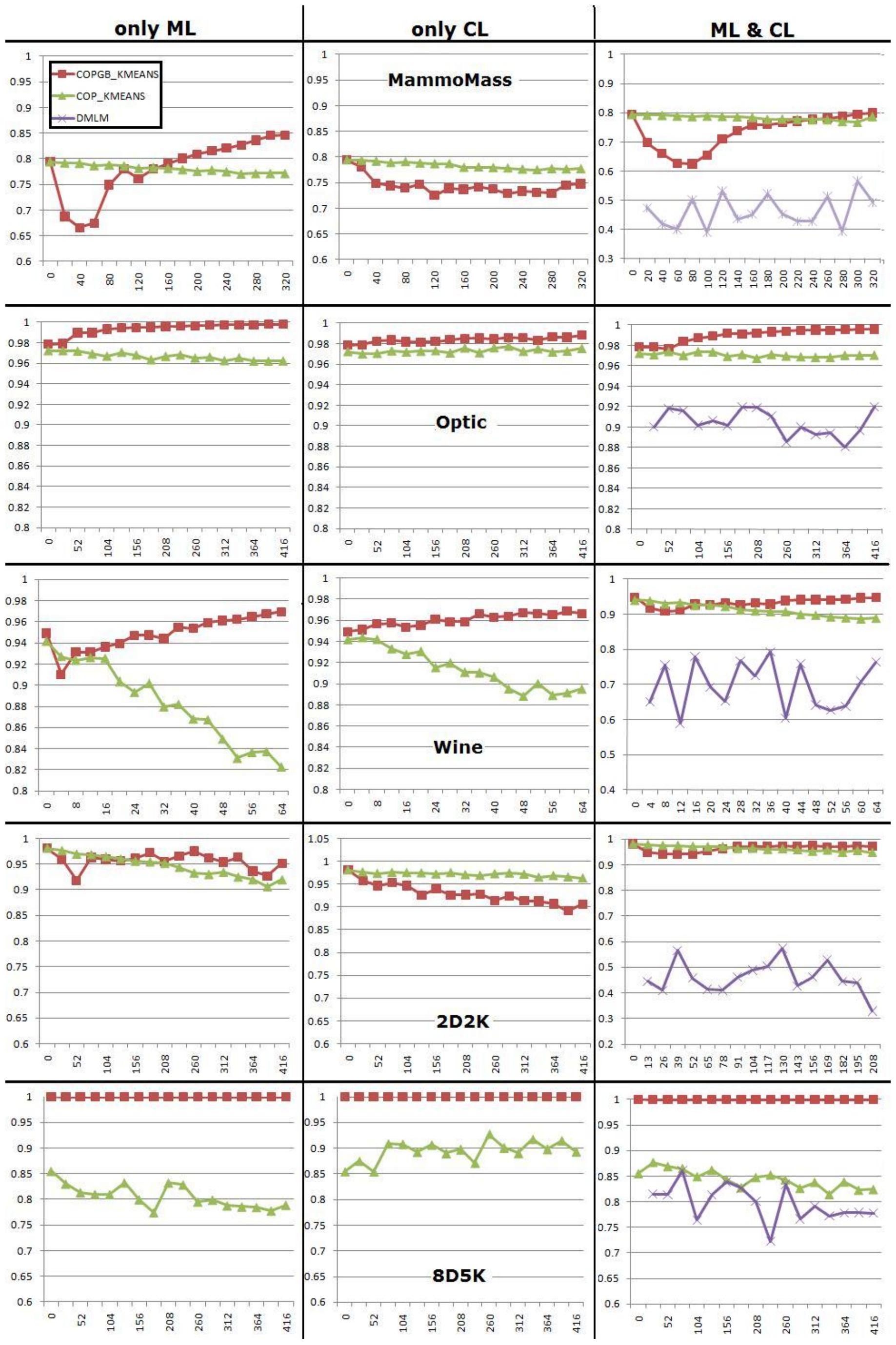


Figure 8.8. Comparison of COP-KM, COPGB-KM and DMLM constrained clustering methods (cont...)

Chapter 9

Constrained Ensemble Clustering

The idea of combining both ensemble and constrained clustering methods is motivated given two main factors. Some datasets present unknown data structure leading to an uncertainty regarding the clustering algorithm to be used. Such uncertainty classifies it as a perfect candidate for the usage of ensemble clustering. This method also works as a way to smooth the final result when different partitions can potentially present dissimilar distributions. Ensemble clustering is also a valid way to improve the final result by gathering correct evidence among all available partitions. On the other hand, it is possible that extra information that can be transformed as constraints to be available. Therefore, constrained clustering presents itself as an indicated approach. Partially labeled data, expected maximum and minimum cluster size, and pairwise linkage pattern relationships are examples of information used by a constrained clustering algorithm. The question arises: "What to do when both unknown data structure and extra information are present simultaneously?". In some cases, the usage in conjunction of both methodologies can be beneficial.

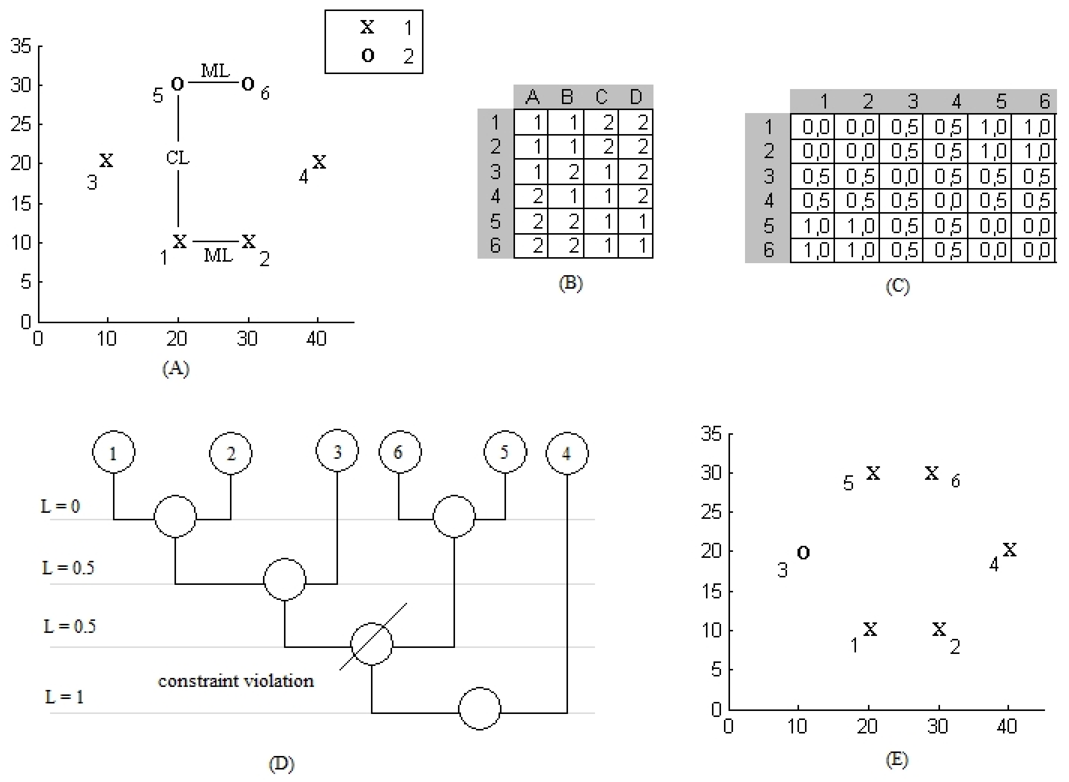
The next logical question that needs to be answered is: "How to combine both approaches?" At first glance, the problem seems to be easily addressed. However, some details need to be taken care of in order to provide a feasible algorithmic realization. In special, attention needs to be paid on investigating the need of considering constraints in both steps of the ensemble clustering process. As presented in Chapter 3, ensemble clustering is a two step process what leads to the inference of four possible scenarios, where constraints are considered not necessarily in both steps. The possibilities are described below.

- **Constraints are considered in both, generation and consensus steps** - In this case, the set of available constraints is used to generate the partitions of the ensemble. Afterwards, the same constraints are accounted for during the combination step;
- **Constraints are considered only by the generation step** - In this case, the set of available constraints is used to generate the partitions of the ensemble, and disregarded by the consensus function;
- **Constraints are considered only by the consensus step** - In this case, the set of available constraints is disregarded during the generation of the ensemble, and used by the consensus function;
- **No constraints are considered in neither, generation nor consensus steps** - This is in fact the general case of ensemble clustering.

The interest arises in the following question: "If all partitions of an ensemble satisfy the constraints, there is still a need to consider them in the combination step?" The answer is definitively positive. However, another side questions can also be addressed, namely, "if the available constraints are not used during the generation step, and later considered only in the consensus step, this would be sufficient? Would the results be compromised in any way?". Both questions are later evaluated and conclusions are drawn about the steps in which constraints must be considered.

In order to understand the reasons why constraints need to be considered in the combination step, a didactic example is shown that constraints can actually be violated. It follows the same combination clustering steps described in Chapter 3, except that a constrained version of K-Means [157] is used to generate the ensemble. It is shown that even if all partitions of an ensemble satisfy the constraints, there is still need of carefully considering the constraints in the combination step in order to avoid violations in the final combined clustering.

Consider the example presented in Figure 9.1. Here, the original data set (A) is used consisting of six patterns to produce four partitions (number K of clusters = 2). In (B), columns represent different partitions and patterns are indexed by rows. Each table cell contains a label assigned to that pattern by the clustering algorithm. All partitions of this ensemble satisfy the two must-link constraints (between patterns 1 and 2, and 5 and 6, respectively) and the cannot-link constraint (between patterns 1 and 5). In (C), the ensemble is used to compute the co-association matrix. The dendrogram produced by the standard single-link (SL) algorithm is shown in (D). Finally, (E) exemplifies an erroneous combination result due to the non-consideration



	A	B	C	D
1	1	1	2	2
2	1	1	2	2
3	1	2	1	2
4	2	1	1	2
5	2	2	1	1
6	2	2	1	1

	1	2	3	4	5	6
1	0,0	0,0	0,5	0,5	1,0	1,0
2	0,0	0,0	0,5	0,5	1,0	1,0
3	0,5	0,5	0,0	0,5	0,5	0,5
4	0,5	0,5	0,5	0,0	0,5	0,5
5	1,0	1,0	0,5	0,5	0,0	0,0
6	1,0	1,0	0,5	0,5	0,0	0,0

Figure 9.1. Misclassification due to non-consideration of constraints in the combination step

of constraints in the combination process. By inspecting the dendrogram, it is easy to see that both must-link constraints are satisfied. When merging the clusters $\{1, 2, 3\}$ and $\{5, 6\}$ in the dendrogram, the resultant cluster clearly violates the cannot-link constraint between patterns 1 and 5. Naturally, this is an example specially designed to present a constraint violation. However, later on this chapter an evaluation using real datasets is presented which the same constraints violation problem happens.

The methods proposed in this chapter assume that given an ensemble of partitions, which satisfies all constraints, the combination algorithm must also consider the same constraints. Otherwise, the constraints may be violated in the final clustering result during the consensus step.

Based on this consideration, the well known clustering combination methods based on evidence accumulation [57] and best one element moves [67] as well as the sum of parwise distances introduced in the last chapter are extended to handle constraints, thus proposing a complete chain of constraint clustering combination.

9.1 Constrained Ensemble Clustering Framework

Constrained ensemble clustering can look very confusing at first glance. It blends two seemly new approaches that by themselves, build up additional complexity over the clustering process. Furthermore, the possibility of considering constraints in only part of the clustering ensemble framework can increase the confusion even further. Perhaps, a simpler way to understand the proposed framework is to realize that it is essentially an extension of ensemble clustering in which extra information converted as constraints is used.

The general model for combination constrained clustering is presented in Figure 9.2. As argued earlier, the set of constraints can be considered in any combination step individually or in both steps. Therefore, the proposed model requires as input, the dataset to be clustered and the set of constraints. For better clarity, the case in which constraints are considered in both steps is described. To simulate the cases that the constraints are considered only during generation or combination, one just disregard the constrained version referring back to the original ensemble clustering step described in Chapter 3.

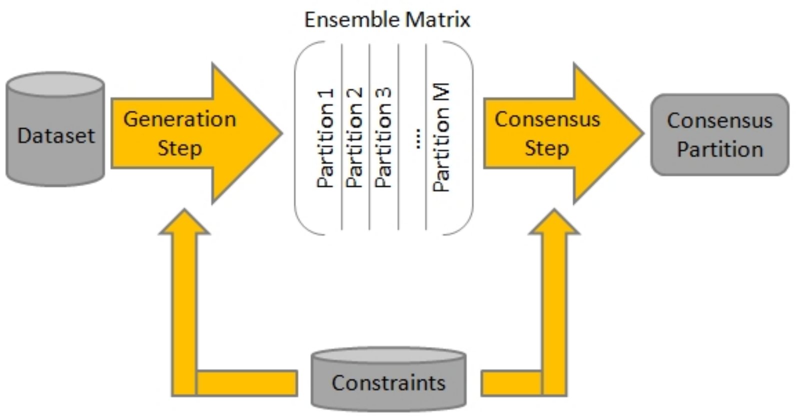


Figure 9.2. General framework for constrained ensemble clustering

Ensemble Generation Step

During the generation step, M partitions are produced and gathered in an ensemble. A special remark needs to be made regarding the consideration of constraints. Instead of applying standard clustering algorithms such as K-Means or spectral clustering, the partitions need to be generated using constrained versions of the desired algorithms. Chapter 8 presents a comprehensive review of the available methods. It is also important to use the same set of constraints in order to ensure coherence among the partitions in the ensemble. Later on, during the combination step, it is expected the same set of constraints to be used as well.

As a pre-processing step, the set of constraints should be evaluated regarding its correctness in order to avoid invalid constraints as well as to infer new constraints based on the existing ones. This can be achieved by computing the transitive closure of the set of constraints, as described in Chapter 3.

The ensemble generation schemes described in Chapter 2 are adopted by the experiments presented in this chapter. More specifically, the ensembles generated using random K and subsets of attributes. Constraints are generated based on the known ground-truth using the method described in [14].

By the end of the generation step, an ensemble of partitions taking into account the constraints is available. The ensemble, together with the same set of constraints used in the generation step is the input required by the constrained consensus step.

Consensus Function Step

To constrain a consensus function can be a trick proposition due to the multitude of combination methods available. However, based on the taxonomy presented in Chapter 3, it is possible to differentiate two main possibilities: a) based on the median partition formulation; and b) based on co-occurrence and based on median partition. Any such consensus function based on those two models can be potentially used. However, it needs to be adapted to work with constraints. Three constrained consensus functions are presented in the next section, based on the co-occurrence, the median partition and the sum of pairwise distances. By the end of the process, a consensus partition is produced. It is supposed to complain with the constraints used during the process by presenting no violations whatsoever.

Even though, the general idea is very simple, it states that the consensus function should work as usual, but before that any label is assignment to a pattern, a constraint violation test should be executed in order to ensure that no invalid assignments occur. The process is able to back-track its original procedure by picking the next best assignment and the process continues. However each case is specific, since the check for constraints violation is dependent of the algorithm's mechanics.

Results Assessment

The results obtained by constrained ensemble clustering algorithms can be assessed by the same criteria described in Chapter 3 regarding general ensemble clustering evaluation. However, special attention needs to be taken regarding the compliance to the original set of constraints. It is expected no constraint to be violated. Revisiting the four possible scenarios, and based on the motivational example presented in the beginning of this chapter, it is clear that in cases which the consensus function is not constrained but the generation step is, violations can occur. Furthermore, if the generation is not constrained but the consensus function is, one could expect no constraints to be violated.

9.2 Proposed Consensus Functions

This section presents three strategies to address the problem of constrained ensemble clustering. The first one is based on the median partition problem. A simulated annealing based algorithm is proposed. The second method is based on the idea of co-occurrence of patterns among the partitions in the ensemble. The original method was proposed by Fred *et al.* [57] in the context of ensemble clustering. It requires an intermediary step in which a co-association matrix needs to be computed. The final method is based on the sum of pairwise distance introduced by this thesis (Chapter 7). It requires, as a pre-processing step, that the pairs of most dissimilar partitions to be determined. The optimization method is also based on simulated annealing. For the purposes of this section, the fully constrained case, in which constraints are considered in both generation and consensus steps, is described. To answer the question regarding the sufficiency of constraints in only one of the two steps, as explored in Section 9.3 of this chapter, the unconstrained steps described in Chapter 3 can be used.

The generation step is a process in which the original data set is provided as input. It outputs an ensemble of M partitions. Additionally, a set of pairwise linkage constraints is provided. Any constrained clustering algorithm can be used during this step. Regarding the number of clusters to be produced, it can be fixed, if the optimal number is known, or decided by the clustering algorithm, if it supports such feature. It is also possible to use an arbitrary random number of target clusters within a range, since the partitions in the ensemble are regarded only as evidence about how the data is structured and not necessarily final results.

For the experiments presented later on this chapter, a constrained version of K-Means is used [157] to produce the ensemble. The constrained version of K-Means is shown in Algorithm 9.5. From this point on, this algorithm will be referenced as cop-KM. The main change to the original version can be found at lines 1 and 3. In 1, the transitive closure over $\mathcal{ML}$ and $\mathcal{CL}$ is computed (see Subsection 8.1.3). This step is required in order to avoid deadlocks during the label's assignment step. Lines $4-6$ implement a conditional statement defining that a pattern can be assigned to

Algorithm 9.5 Constrained K-Means algorithm (cop-KM)

```
Input: $D$ : Data set
       $\mathcal{ML}$: Set of must-link constraints
       $\mathcal{CL}$: Set of cannot-link constraints
       $K$ : number of clusters
Output: a partition of $D$ presenting no constraint's violations

01. compute the transitive closure over $\mathcal{ML}$ and $\mathcal{CL}$
02. randomly initialize the cluster centers by $C_1, \cdots, C_K$
03. $\forall\ d_i \in D$
04.   if no constraint is violated
05.     assign $d_i$ to the closest cluster center $C_j$
06.   end
07. end
08. $\forall\ C_j$
09.   compute the new cluster center by averaging all $d_i \in C_j$
10. end
11. GOTO 2
```

the closest cluster if and only if no constraint is violated, see [34, 155] for further details of the constraint violation test and the transitive closure computation. If the closest cluster cannot be chosen as target cluster, the algorithm proceeds by checking all remaining clusters until no one are left. If no allowable cluster is found, it returns an empty label for that pattern.

9.2.1 Median Partition

The first constrained ensemble clustering method proposed is based on the computation of the median partition. Since the median partition problem is $\mathcal{NP}$-Complete for many reasonable distance function, as described in Chapter 3, it is necessary rely on approximate solutions. There is a number of heuristics proposed in the literature (see [67] for 6 heuristics) for the computation of approximated solutions. This section presents an approximated solution based on the simulated annealing algorithm.

Figure 9.3 shows the fluxogram for the constrained consensus function based on the simulated annealing. It receives as input the ensemble of partitions $\mathbb{P}$ and the sets of must-links $\mathcal{ML}$ and cannot-links $\mathcal{CL}$. The first step refers to the selection of

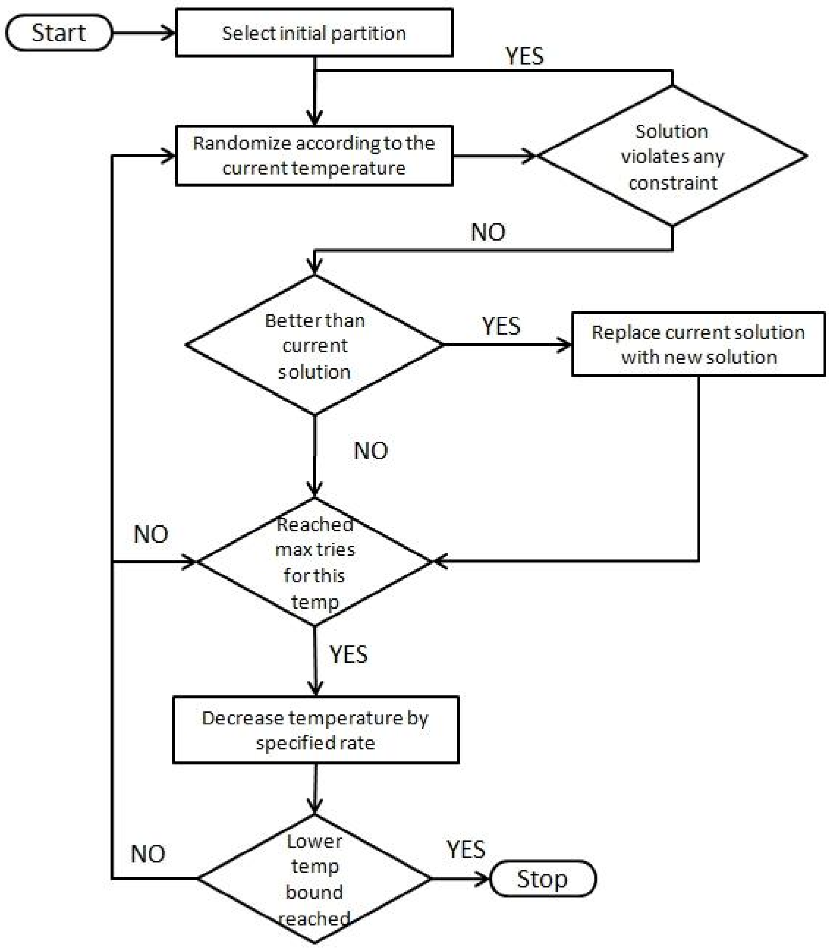

Start
Select initial partition
YES
Randomize according to the current temperature
Solution violates any constraint
NO
Better than current solution
YES
Replace current solution with new solution
NO
NO
Reached max tries for this temp
YES
Decrease temperature by specified rate
NO
Lower temp bound reached
YES
Stop

Figure 9.3. Constrained *BOEM* simulated annealing based algorithm

an initial consensus partition[1]. In the experiments presented later, the set median P^* is selected to be the initial partition. A starting temperature is defined as an algorithm's parameter and, based on that, a randomization of the partition P* is executed, producing a new candidate partition $\tilde{P}$. The partition $\tilde{P}$ is then evaluated by means of the computation of its SoD between $\mathbb{P}$ and $\tilde{P}$. If the current partition presents a smaller SoD compared to the original candidate partition, the result is retained otherwise, a new the randomization is generated. The temperature is lowered and the process continues until the maximum number of tries for the current temperature is reached. If the lower bound temperature is reached the process stops and $\tilde{P}$ is returned, otherwise a new randomization of $\tilde{P}$ is produced and the process starts all over again.

[1]The set median as described in Chapter 2 is simply the partition belonging to the ensemble $\mathbb{P}$ with presents the smaller SoD in relation to all partitions in the ensemble. It is also known as the BoK or best of K method.

9.2.2 Evidence Accumulation

The second constrained ensemble clustering method is based on the well known ensemble clustering algorithm proposed by Fred *et al.* [57]. Its motivational idea is that existing evidence, distributed among the partitions of the ensemble can be used to infer the consensus partition.

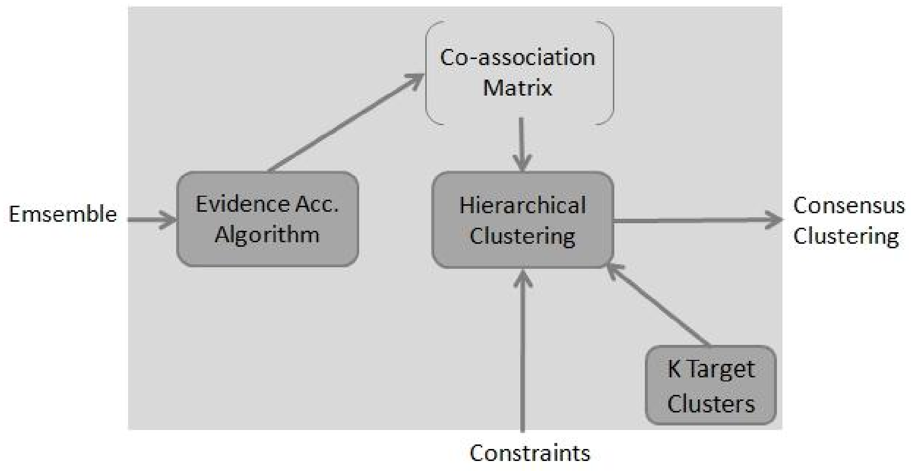


Figure 9.4. Schematics for the combination step of the constrained combination clustering

Given the ensemble of partitions generated using constrained clustering algorithms, the method follows by computing a co-occurrence matrix. This is done by means of Equation (3.8). Note that for the computation of the co-association matrix, there is no need to consider again the constraints, since the constrained clustering algorithms used during the generation step will ensure no violation of the constraints will be presented in the partitions.

Differently from most of the consensus functions presented in Chapter 3, co-occurrence based methods do not take the ensemble of partitions as input. Instead, the co-occurrence matrix is used. It is regarded as a new feature's space. Since the input is in the form of a similarity matrix, a consensus function able to work over this kind of input data is required. As proposed in [57], a hierarchical clustering algorithm is used in order to produce the final consensual partition. A constrained version of hierarchical single-link algorithm is used as consensus function. The algorithm for constrained agglomerative single-link (see Algorithm (9.6)) is adapted from [38]. A similarity matrix SM and two sets of $\mathcal{ML}$ and $\mathcal{CL}$ constraints are the input required. It starts by assigning all patterns to singleton clusters, namely, $C_1, \cdots, C_N$. The only change to the original single link algorithm refers to the

Algorithm 9.6 Constrained agglomerative algorithm (cop-SL)

```
Input: $SM$ : Dataset
       $\mathcal{ML}$: Set of must-link constraints
       $\mathcal{CL}$: Set of cannot-link constraints
Output: a dendrogram

1. compute the transitive closure over $\mathcal{ML}$ and $\mathcal{CL}$
2. repeat steps 3-12
3.     find the minimum entry $(x_l, x_m) \in SM$
4.     if merging $x_l$ and $x_m$ violate any constraint in $\mathcal{CL}$
5.        find the next minimum entry in $SM$
6.     else
7.        Merge$(x_l, x_m)$ and update $SM$
8.     end
9.     if there are no more clusters to be merged
10.       return the dendrogram
11.    end
12. end
```

introduction of a constraint violation test before two clusters selected for merging. The algorithm must stop if no there are no more clusters to be merged without constraint's violation. In this case, the final reached state of the dendrogram also represents the minimum obtainable solution in the presence of constraints. In [38] a test is presented to verify the minimum number of clusters obtainable in the presence of $\mathcal{CL}$ constraints. The first line finds the minimum entry in SM and merges the clusters it refers to if no $\mathcal{CL}$ constraint is violated. It selects the next minimum on SM, otherwise.

Since must-link constraints are transitive, it is also possible to compute the transitive closure for the $\mathcal{ML}$ set as a pre-processing step (see [38]). However, this is not the case. Since the computation of the co-association matrix ensures maximum similarity (i.e. 1) to any pair of $\mathcal{ML}$ constrained patterns these patterns will be merged during the initial iterations of cop-SL. This is the reason why algorithm cop-SL has no explicit handling for must-links.

9.2.3 Sum of Pairwise Distances

The final constrained ensemble clustering method is based on the sum of pairwise distances formulation presented in Chapter 7. For the ensemble generation step, no special remark needs to be made. The consensus function works like the one presented by the fluxogram given in Figure 7.4. However, a special attention needs to be paid regarding the consideration of constraints. Figure 9.5 shows the fluxogram for the simulated annealing based $SoPD$ constrained consensus function.

There are two major processes that take place originally belonging to the simulated annealing method and two introduced by the $SoPD$ formulation. Additionally, a check regarding constraint's violation is executed in order to ensure no violations. At the beginning, a candidate initial partition is selected. As stated before, any simple consensus function is suitable for this step. In the experiments presented later in this chapter, the set median is used to produce the initial candidate partition $\tilde{P}$.

The general simulated annealing method states that for each temperature, a number of cycles will take place. This number can be user defined or specified by the algorithm's designer. As a cycle runs, the inputs are randomized. Only randomized partitions which produce a better set of inputs are retained. In the case of ensemble clustering, the randomization can be achieved by changing the label of a random pattern within a given label's range. Just after a new randomization is generated, the test for constraints violation is executed. If any constraint is violated by the perturbation induced during the previous step, it is discarded and a new perturbation is generated.

Following the fluxogram, a test to assess if the newly candidate partition is more accurate than the current one is executed. This is based on the $SoPD$ value returned by Equation (7.4). It this value is smaller than the one computed for $\tilde{P}$, the partition P' is attributed to $\tilde{P}$. Once the specified number of training cycles has been completed, the temperature can be lowered. Once the temperature is lowered, it is determined whether or not the temperature has reached the lowest temperature allowed. If the temperature is not lower than the lowest temperature allowed, then the temperature is decreased and another cycle of randomized partition generation takes place. At this point, a new set of most dissimilar pairs of partitions will be computed as described in Section 7.6. It receives as input the ensemble of partitions ($\mathbb{P}$) and the current candidate partition P'. The output of this process is a set of pairs of partitions $(P_i, Q_i), i = 1 \cdots M/2$, where M is the number of partitions in $\mathbb{P}$ and $P_i, Q_i \in \mathbb{P}$ such that the distance between them is maximal. The process continues by computing the $SoPD$ according to the Equation (7.4). The pairings based on $\tilde{P}$ and the corresponding $SoPD$ is computed. This value will be used to assess if the progressively proposed candidate partitions P' achieve better results than $\tilde{P}$.

If the temperature is lower than the lowest temperature allowed, the simulated annealing algorithm will end by attributing $\tilde{P}$ as consensus partition.

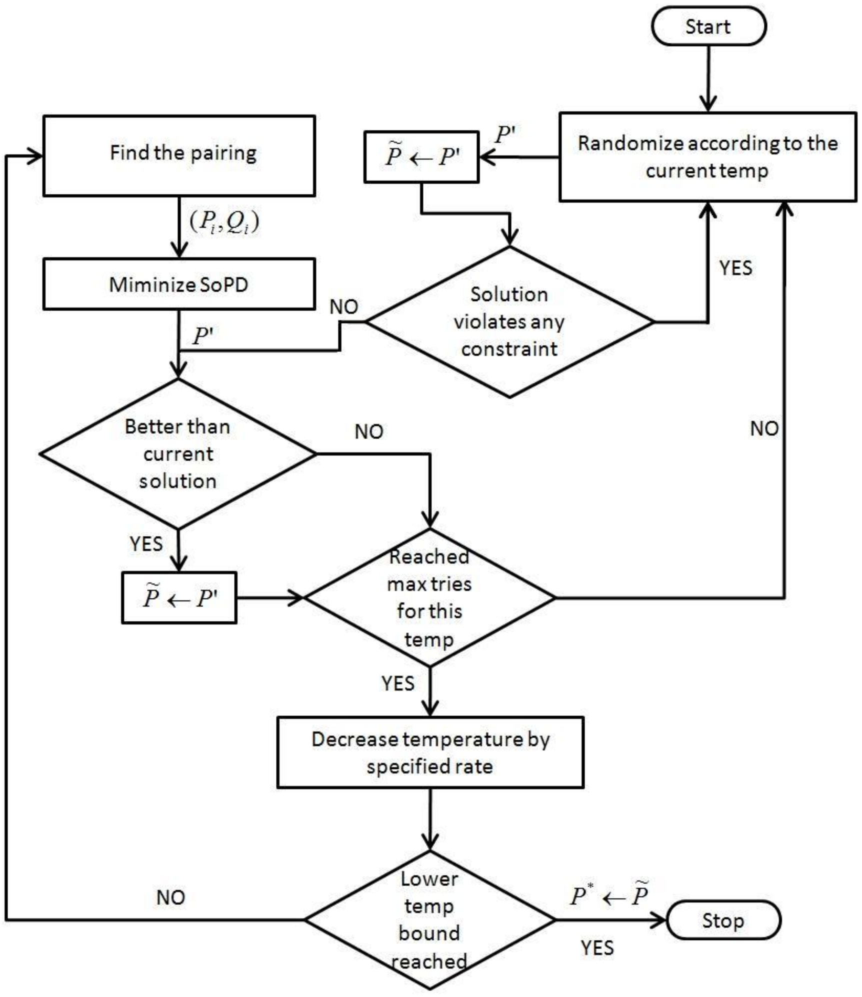

Start
Randomize according to the current temp
P'
$\tilde{P} \leftarrow P'$
Find the pairing
(P_i, Q_i)
Miminize SoPD
P'
NO
Solution violates any constraint
YES
Better than current solution
NO
NO
YES
$\tilde{P} \leftarrow P'$
Reached max tries for this temp
YES
Decrease temperature by specified rate
NO
Lower temp bound reached
$P^* \leftarrow \tilde{P}$
YES
Stop

Figure 9.5. Fluxogram for the simulated annealing $SoPD$ consensus function

9.3 Experimental Results

The combination of constrained and ensemble clustering into a single clustering solution poses a fundamental question that needs to be addressed. In which steps of the ensemble clustering framework is necessary/sufficient to consider constraints? In order to answer this question as well as to evaluate the results accuracy, a series of experiments are devised.

Firstly, it is investigated if the non-consideration of constraints in the consensus (final) step leads to constraint's violations. By considering constraints in both steps, or only during the consensus step, there is no need to worry about constraints being violated in the consensus partition. This is due the premise that any constrained clustering algorithm or constrained functions should comply with all the constraints in the $\mathcal{ML}$ and $\mathcal{CL}$ sets. However, for the case in which constraints are considered only during the generation step violations can occur, since, the consensus function proceeds as if no prior knowledge about the data existed. In fact, Figure 9.6 shows an experiment in which constraints are considered only during the generation step. For this experiment, the number of constraints grows incrementally. Half the constraints is must-links and half cannot-links. The number of violations due to non-consideration of constraints in the combination step increases for both synthetic datasets (top-left) and UCI datasets (top-right, bottom-left and bottom-right). The graphic shows that the number of constraints violated (y-axis) grows with the number of constraints being considered (x-axis). The accuracy of the consensus result also diminishes with the increase number of constraints being violated, contrary to what usually occurs in constrained clustering in which the accuracy tends to increase with the number of constraints used.

Despite of the fact that the accuracy of each individual partition in the constrained ensemble presents improved result due to the introduction of pairwise linkage constraints, the same does not necessarily happens in which these constraints are not considered in the consensus function. In fact, for most cases, the results can be worst. The results decrease in accuracy can be corroborated in the experiment that follows.

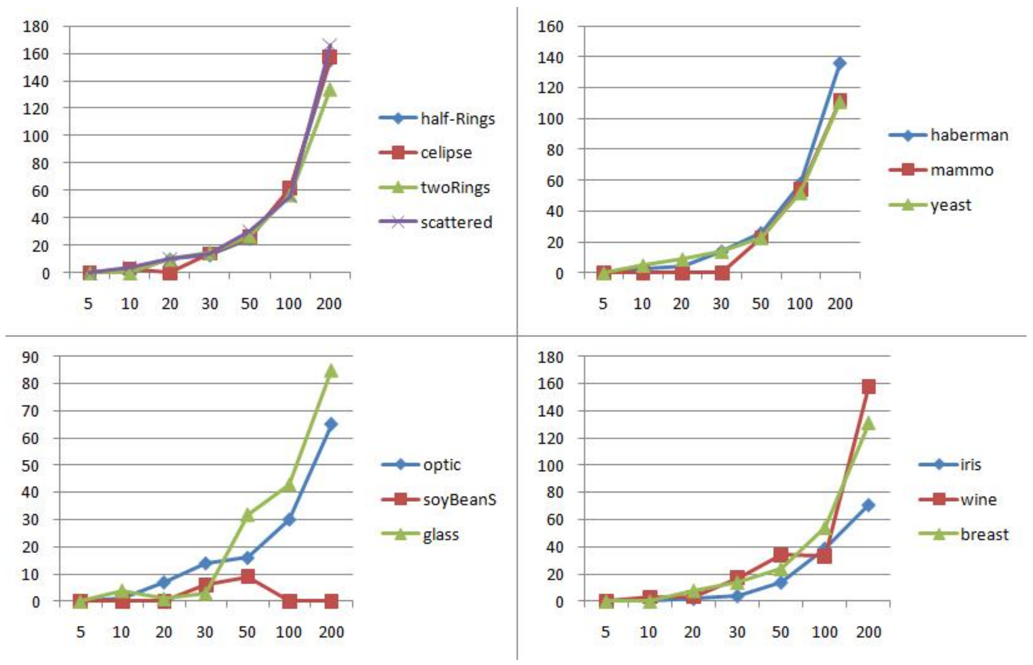


Figure 9.6. Number violations due to non-consideration of constraints in the combination step

The second envisioned experiment is a comparison of the impact of considering constraints, as stated before, in three different ways:

a) only during the generation step;

b) only during the consensus step;

x) during both steps.

The experiment is divided in two parts. In part one, the 6 toy datasets are evaluated. The results are listed in Tables 9.1, 9.2, and 9.3. Part two evaluates the results of the UCI-Irwine datasets, listed in Tables 9.4, 9.5, and 9.6. The initial column of all tables is the average result of the ensemble clustering methods presented in Table 3.2 in which no constraints are considered. It works as a control, or reference to the case in which constraints are not considered in any of the ensemble clustering steps. The second column is the average result of 20 runs of cop-KM algorithm. Those 20 partitions are used to compose the constrained clustering ensemble used in these tests.The number of constraints was set to 10% of the total number of patterns in each dataset for each type of pairwise constraints, namely must-links and cannot-links. The impact of different amount of constraints was studied in Chapter 8. The four remaining columns in the tables list the results of different consensus functions namely, two instances of evidence accumulation based methods (EAC_SL and EAC_AL), an instance of median partition based methods ($BOEM$) and finally, an instance of sum of pairwise distance using simulated annealing as optimization function ($SoPD_{sa}$).

Table 9.1 shows the results for the worst case, in which constraints are considered during the generation but not in the consensus step. Those are results similar to the case presented in Figure 9.6 in which constraint violations do occur in the final consensus partition. It is possible to identify that for most cases the results obtained by EAC_SL, EAC_AL, $BOEM$ and $SoPD_{sa}$ (columns 4-7) consensus functions are in general worst or slightly better than the ones obtained by the average of ensemble clustering methods (CE) which do not consider constraints in any step (column 1). The same happens when a comparison is made against the average results of a simple constrained clustering result (column 1). It is safe to conclude that by considering the constraints only in the generation step do not pay off the additional computational time required by the introduction of constraints. This can be justified by the fact that in the generation step, in which the set of constraints is considered, the normal way that the clustering algorithm works is constrained, guided to accommodate the extra knowledge encoded as linkage constraints. By disregarding such knowledge in the subsequent step (consensus step) the algorithm, following its general structure, produces contradictory results to the extra information available, having a final impact in the overall result.

Table 9.1. Results for constrained generation, unconstrained consensus on toy datasets

Dataset	CE	cop-KM	EAC_SL	EAC_AL	$BOEM$	$SoPD_{sa}$
8D5K	11.04	10.04	60.00	60.00	26.94	24.11
2D2K	8.74	3.94	48.20	4.70	6.40	54.40
celipsoid	30.51	28.18	31.11	27.11	28.00	62.22
twoRings	47.67	42.54	72.93	67.40	47.24	71.55
scattered	44.48	44.92	44.70	43.94	46.21	48.48
halfrings	19.49	13.75	40.89	11.90	12.27	66.91

The results obtained by the consideration of constraint only during the consensus step are presented in Table 9.2. A different sort of problem is observed, although no constraints are violated by the consensus partition. It does not come as a surprise, since the constraints are considered in the final step, and it was postulated earlier that either generation or consensus algorithms ensures no constraint violation. For some cases, such as 8D5K, 2D2K, and halfrings, $SoPD_{sa}$ scored better than CE, and better or equal to cop-KM. EAC_SL and EAC_AL scored better for the scattered dataset. However, there is no guarantee that the constrained ensemble clustering methods will always score better when constraints are considered only during the consensus step, due to the fact that the extra information is ignored during the generation step. By doing so, it can lead to erroneous assignments during the consensus step.

Table 9.2. Results for unconstrained generation, constrained consensus on toy datasets

Dataset	CE	cop-KM	EAC_SL	EAC_AL	$BOEM$	$SoPD_{sa}$
8D5K	11.04	10.04	23.17	23.17	20.32	0.00
2D2K	8.74	3.94	45.10	45.90	6.40	1.90
celipsoid	30.51	28.18	45.33	45.33	27.11	35.11
twoRings	47.67	42.54	46.41	44.75	47.24	69.06
scattered	44.48	44.92	27.27	27.27	46.21	57.58
halfrings	19.49	13.75	86.99	86.99	44.98	13.75

For the case in which constraints are considered in both step (Table 9.3) a considerable improvement can be noticed. Firstly, $SoPD_{sa}$ always scores better than both CE and cop-KM. $BOEM$ always scores better than CE, but it shows a slightly decreasing in performance for the twoRings dataset if compared to the average cop-KM. The evidence accumulation methods (columns 3 and 4) are however not unanimous. For 8D5K the results are actually worst than CE and cop-KM. In 2D2K EAC_SL is able to achieve a perfect classification (0% of errors) but EAC_AL scored very badly. For halfrings, the results are worst and the remainder datasets produced average results in which better or similar scores are achieved. It is possible to conclude that the evidence accumulation methods do not share a unified performance. Although, these methods could be able of performing really well, they are unreliable for some cases.

Table 9.3. Results for constrained generation, constrained consensus on toy datasets

Dataset	CE	cop-KM	EAC_SL	EAC_AL	$BOEM$	$SoPD_{sa}$
8D5K	11.04	10.04	23.17	23.17	0.00	0.00
2D2K	8.74	3.94	0.00	45.10	1.90	1.70
celipsoid	30.51	28.18	31.11	24.44	28.00	27.11
twoRings	47.67	42.54	45.30	46.96	47.51	42.51
scattered	44.48	44.92	27.27	31.82	43.18	43.18
halfrings	19.49	13.75	44.98	44.98	12.27	11.38

Table 9.4 shows the results for the UCI-Irvine datasets, in which constraints are considered during the generation but not in the consensus step. Those results are similar to the case presented in Figure 9.6 in which constraint violations do

occur in the final consensus partition. EAC_SL performs worst in 88% of the cases in comparison to CE and 84% compared to cop-KM. EAC_AL scores 84% worst compared to CE and 88% worst than cop-KM. $BOEM$ is the best consensus function in this case. It scores in 58% of the cases worst than CE and 60% worst than cop-KM. Finally, $SoPD_{sa}$ scores in 88% of the cases worst than CE and in 92% of the cases worst than cop-KM. As it can be concluded, it is better to use only constrained clustering or ensemble clustering if constraints are considered only during the generation.

Table 9.4. Results for constrained generation, unconstrained consensus on UCI-Irvine datasets

Dataset	CE	cop-KM	EAC_SL	EAC_AL	$BOEM$	$SoPD_{sa}$
balance	39.40	38.66	52.96	52.96	41.76	56.53
breast	13.68	5.30	66.18	68.81	34.99	4.69
control	39.29	38.95	66.67	66.67	33.83	41.17
ecoli	48.68	47.29	55.36	55.36	48.51	46.44
glass	49.64	57.20	52.34	62.62	60.75	67.12
haberman	30.32	37.55	50.00	49.67	50.00	60.51
heart	39.42	46.11	44.07	47.41	49.26	65.94
ionosphere	32.26	30.37	35.33	31.05	31.62	70.94
iris	11.12	11.47	33.33	33.33	6.67	40.03
lung	27.55	32.59	33.33	22.22	11.11	74.02
mammo	23.96	18.61	48.31	48.31	17.35	58.29
optic	26.98	24.15	88.80	88.80	15.30	26.18
parkinsons	31.49	25.85	26.67	26.67	26.67	48.72
post_op	47.90	55.86	29.89	48.28	58.62	57.44
protein	51.67	43.28	71.55	57.76	34.48	43.82
segmentation	41.48	49.48	84.76	71.43	48.10	56.66
sonar	44.97	43.75	73.08	72.60	48.08	73.08
soyBeanS	27.90	28.09	40.43	40.43	21.28	46.17
spect	43.24	44.31	41.20	44.19	46.44	46.81
spectf	37.93	44.46	89.51	72.66	42.70	69.66
taeval	53.64	62.45	63.58	63.58	64.90	78.29
tic-tac-toe	46.88	44.52	68.06	67.75	45.30	67.95
transfusion	34.89	23.92	24.47	24.47	24.47	77.41
wine	7.92	10.34	33.71	30.90	4.49	37.63
yeast	57.05	66.46	68.13	68.67	69.41	72.99

The results obtained by the consideration of constraint only during the consensus step presented in Table 9.5. EAC_SL performs worst in 52% of the cases compared to both CE and cop-KM. EAC_AL scores 56% worst compared to CE and 48% worst than cop-KM. $BOEM$ is the best consensus function again. It scores in only 44% of the cases worst than CE and 60% worst than cop-KM. Finally, $SoPD_{sa}$ scores in 72% of the cases worst than CE and in 72% of the cases worst than cop-KM. As it can concluded, when constrains are considered only during the consensus step, a better performance is achieved in comparison to the case in which constraints are used only during the generation step. However, the computed percentages still do not validate the usage of constrained ensemble clustering as a viable option to achieve better results compared to any of the two approaches considered individually.

Table 9.5. Results for unconstrained generation, constrained consensus on UCI-Irvine datasets

Dataset	CE	cop-KM	EAC_SL	EAC_AL	$BOEM$	$SoPD_{sa}$
balance	39.40	38.66	47.20	40.96	53.44	53.37
breast	13.68	5.30	34.26	4.25	5.27	36.18
control	39.29	38.95	45.83	42.67	33.83	41.67
ecoli	48.68	47.29	45.83	46.73	48.51	44.90
glass	49.64	57.20	45.79	57.48	60.75	66.86
haberman	30.32	37.55	26.80	60.78	49.67	50.00
heart	39.42	46.11	47.41	47.41	48.89	65.92
ionosphere	32.26	30.37	23.08	32.19	31.62	73.38
iris	11.12	11.47	6.67	6.67	6.67	6.67
lung	27.55	32.59	18.52	7.41	11.11	74.06
mammo	23.96	18.61	51.81	51.81	17.35	52.22
optic	26.98	24.15	55.90	40.40	22.70	23.19
parkinsons	31.49	25.85	26.67	26.67	26.67	26.63
post_op	47.90	55.86	45.98	51.72	58.62	52.85
protein	51.67	43.28	55.17	55.17	45.69	45.12
segmentation	41.48	49.48	44.29	43.81	48.10	44.86
sonar	44.97	43.75	48.08	48.08	48.08	75.48
soyBeanS	27.90	28.09	31.91	8.51	29.79	29.79
spect	43.24	44.31	21.35	21.35	46.44	59.95
spectf	37.93	44.46	42.32	42.70	42.70	42.11
taeval	53.64	62.45	64.24	56.29	64.90	62.12
tic-tac-toe	46.88	44.52	34.55	34.34	45.72	66.18
transfusion	34.89	23.92	24.47	24.47	24.47	24.60
wine	7.92	10.34	4.49	4.49	4.49	4.49
yeast	57.05	66.46	60.78	60.78	53.30	69.01

Table 9.6 shows the results for the UCI-Irvine datasets in which constraints are

considered in all the framework steps namely, generation and consensus. EAC_SL performs better in 52% of the cases compared to CE and 60% of the cases compared to cop-KM. EAC_AL scores 56% better compared to CE and 76% better than cop-KM. $BOEM$ scores in only 88% of the cases better than CE and cop-KM. Finally, $SoPD_{sa}$ scores in 86% of the cases better than CE and in 86% of the cases better than cop-KM. As it can concluded, the consideration of constraints in both steps of the constrained ensemble clustering framework leads to the best possible outcome. It is important to notice that in some cases such as the breast and haberman datasets the improvement is impressive, in which a perfect match is achieved by all constrained consensus functions. It is also noticed the leap in improvement achieved for the wine dataset, in which almost 100% of improvement compared to CE is observed and a lower but still considerable improvement experienced by $BOEM$ although EAC_SL and EAC_AL fail to achieve better than CE. However, all constrained ensemble clustering methods succeed in performing better than cop-KM in this case. As a final remark, it is important to point out that only 10% of the total number of patterns is constrained. This percentage is selected since it seems reasonable to inspect 10% of the data, manually. The results, however, can be greatly improved if more constraints are considered. This is possible in cases of automatic constraints generation in applications such as image segmentation.

Table 9.6. Results for constrained generation, constrained consensus on UCI-Irvine datasets

Dataset	CE	cop-KM	EAC_SL	EAC_AL	$BOEM$	$SoPD_{sa}$
balance	39.40	38.66	21.92	38.24	42.08	24.02
breast	13.68	5.30	0.00	0.00	0.00	0.00
control	39.29	38.95	44.67	38.83	30.83	31.09
ecoli	48.68	47.29	25.00	45.83	40.77	43.45
glass	49.64	57.20	54.21	54.21	46.73	46.89
haberman	30.32	37.55	0.00	0.00	0.00	0.00
heart	39.42	46.11	45.93	46.67	40.74	40.73
ionosphere	32.26	30.37	35.90	40.66	29.06	29.06
iris	11.12	11.47	36.67	66.67	4.00	4.00
lung	27.55	32.59	11.11	7.41	19.63	19.64
mammo	23.96	18.61	40.84	46.39	20.72	20.68
optic	26.98	24.15	52.00	17.20	15.30	28.37
parkinsons	31.49	25.85	13.33	14.87	24.62	24.58
post_op	47.90	55.86	33.33	52.87	36.78	37.11
protein	51.67	43.28	43.97	25.00	33.62	47.40
segmentation	41.48	49.48	44.29	33.33	45.71	45.68
sonar	44.97	43.75	40.38	69.71	43.27	43.19
soyBeanS	27.90	28.09	21.28	21.28	21.28	21.28
spect	43.24	44.31	10.49	24.72	32.21	28.71
spectf	37.93	44.46	20.97	42.32	34.08	34.08
taeval	53.64	62.45	62.91	54.30	52.98	52.98
tic-tac-toe	46.88	44.52	25.37	52.82	39.14	39.14
transfusion	34.89	23.92	38.10	16.58	26.07	26.07
wine	7.92	10.34	23.03	3.37	3.37	3.37
yeast	57.05	66.46	60.34	61.80	44.25	32.08

Part IV

Fiber Segmentation

Chapter 10

DTI Fundamentals

The magnetic resonance phenomenon was discovered by Felix Bloch in 1946. It wasn't until 1972, the first generated image was produced, and in 1974 the first MRI of a living creature. Since then, the necessary technology to apply the MRI in clinical environments has considerably evolved opening a brand new field for human body exploration. MRI is important from the clinical perspective since it is a non-invasive imaging method. Additionally, it imposes no exposure to radiation as e.g. X-Rays. A good theoretical review about MRI can be found in [74].

The first big development after the introduction of the traditional MRI (T1 and T2) was the development of fMRI - functional MRI - a kind of MR-scan that measures the burning of ATPs. It allows the visualization of activated regions in the human brain. The ability to measure the concentration of hydrogen in a given point of the space provided by MRI allows the creation of detailed static volumetric images containing anatomical data (e.g. T1, T2, MPRage, Flair) or detailed dynamic volumetric movies presented the activated regions of a living subject presented with some external stimulus. With the development of DWI, it became possible to inspect the organization of fibrous tissues such as the human brain or muscles, otherwise, it would appear as homogeneous regions in other scanning protocols.

In the past two decades, some physicians [12] come up with the idea to combine the Bloch resonance magnetization equation with the diffusion equation. By doing so, a whole new field in imaging was opened. Taking advantage of an(isotropic) behavior of water molecules present in living tissues, opens the possibility to study how fibrous tissues are organized. By measuring the concentration of water molecules in a given point of a Cartesian 3D space, it is also able to measure the quantity of Brownian motion, or better describing, the amount of average Brownian motion for a given voxel of $XX \times YY \times ZZ$ arbitrary dimensions. The water molecules display a number of interesting properties that makes it to be the ideal choice as standard atom to perform MR-scans. In the human body, it is already well known that different tissues present different apparent diffusion coefficients (ADC). An example can be found by examining tissues like the gray matter, fat, or in the Cerebro-spinal fluid where the ADC is mainly isotropic and it can be taken as a constant to all the media. It is possible to represent the ADC by a single scalar. However, in other tissues, more specifically fibrous tissues, it is also already well known that the Brownian motion is restricted on more specifically directed or oriented among the direction of the fibbers.

The DWI protocol is comprised of an N dimensional object containing the apparent diffusivity of water molecules. The measures are taken usually in 6 independent diffusion directions in each voxel, although there are newer scanning devices which are able to sample more directions. The evaluation of such 6th dimensional data can be difficult, since they are encoded in 6 series of images. It is common to post-processing the data to generate a tensor volume, where each 6th dimensional voxel is combined using numerical methods to produce a tensor. This tensor can be interpreted as a probability movement distribution of the water molecules to behave from its center to the voxel periphery.

The main focus of this chapter is to review the concepts underlying the MR-DWI protocol and to settle the foundation needed in order to motivate Chapter 11 refers to fiber segmentation.

10.1 Diffusion Tensors

Diffusion imaging only measures ADCs in a given direction oriented by the scanner, and in tissues, this orientation is often arbitrary. The solution to this problem is to acquire more than one scan. Figure 10.1 shows a section of a DWI scan in which the control image is shown together with the usually six diffusion directions. At least six diffusion directions are needed, in order to calculate a tensor or diffusion tensor representing the tendency of diffusion in a given voxel.

MR-DWI[1] stands for magnetic resonance diffusion weighted imaging. Weighting in MRI, is the contrast seen in resulting images. Diffusion weighting, therefore, shows as contrast water diffusion. Water diffusion is the name given to the random movement of molecules, also known as Brownian motion. DWI produces images

[1]DWI is also known as DTI or diffusion tensor images since any practical application based on DWI requires the computation of the diffusion tensors volume.

whose contrast indicates the amount of water diffusion in a given voxel. It works by applying two gradients to the magnetic field conventionally used in MRI. The first one is called the dephasing gradient and the second is the rephasing gradient. Both gradients have the same strength and direction, but opposed magnitude.

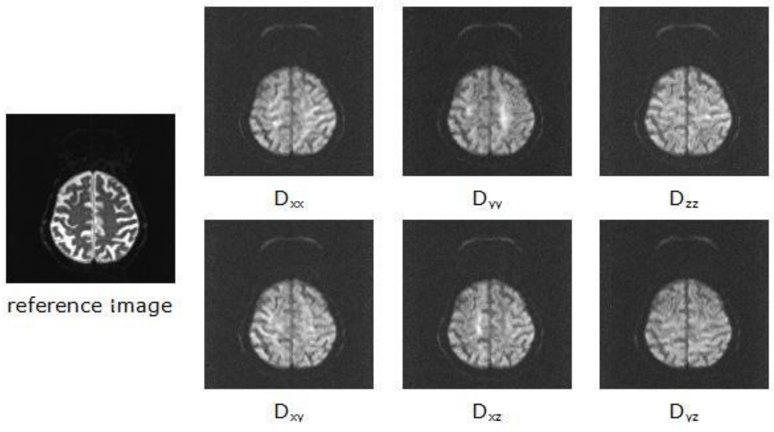


Figure 10.1. Control and DWI images

The measured effect after the gradient application depends on the movement of water: if a water molecule moves approximately the same amount in all directions, the net value measured be 0 (zero), i.e., one gradient nulls the other; but if a water molecule moves preferentially along a specific direction, then the net value of the gradients on this particular molecule will not be zero. Thus, the water movement along the gradient direction can be measured, and the value measured is known as a signal.

There are many parameters characterizing the magnetic field. The most important are the gradient directions and the radio frequency (RF) pulses. These parameters are usually gathered in a constant b defined as:

$$b = \gamma^2 G^2 \delta^2 (\Delta - \frac{\delta}{3}) \tag{10.1}$$

The equation that relates signals measured with gradients and a signal with no gradient applied is:

$$\frac{S}{S_0} = e^{-bD} \tag{10.2}$$

Here, S is the signal with gradients and S_0 with no gradients. D is the amount of water diffusion along the gradient direction, i.e., it is the diffusion coefficient.

The signal S_0 is obtained through the same process as S, except that no gradients are applied. Usually S_0 is obtained with $b = 0$. Indeed, if $b = 0$, then:

$$\frac{S}{S_0} = e^0 = 1 \tag{10.3}$$

therefore $S = S_0$ and the previous equation still holds.

The ratio $\frac{S}{S_0}$ is sometimes called attenuation and its symbol is A. The above equation may then be written as the signal equation:

$$A = e^{-bD} \tag{10.4}$$

It is noteworthy that these equations have been used, up to now, with a single gradient. It can be understood as the water diffusion measurements alongside a single direction.

As discussed earlier, water molecules in human tissue hardly diffuses in an isotropic way. The amount of water diffusion is not the same in all directions. This type of diffusion is called anisotropic diffusion. Figure 10.1 shows diffusion weighted images taken from the brain with gradients along the x, y and z axes. It is clear from these images that water diffusion in brain is anisotropic in many regions, more specifically, in the regions comprehending the white matter. To understand what these images really mean, it is required to comprehend entirely the ideas of diffusion, and gradients measurement.

If a drop of water falls upon a plain absorbing surface, like a soft cloth, the water will be slowly absorbed by the cloth. After some time, the water will mostly have drawn a circle or an ellipse on the surface as it was absorbed. It could be said that water diffused through the cloth, and that the shape drawn by the water is a function of the way the cloth allows the water to diffuse. On most surfaces water will probably spread circularly, but on certain clothes it might as well spread drawing an ellipse. On some surfaces, such as wood, water wouldn't spread at all. They would keep their original position for a very long time. Not only surfaces determine the way water spreads, but also how fast it spreads. This property, namely "diffusion" can be described, by an ellipse. Its shape would show how water spreads along each direction on the surface, and its size would tell how far it spreads. If water spread equally on all directions, this ellipse would become a circle.

Any ellipse centered at the origin needs exactly three parameters to be fully specified. These parameters could be, for example, the length of its two axes (along the x and y-axis) and an angle determining rotation. But a more useful way of describing ellipses is by using 2×2 symmetric matrices with some special properties. Such a matrix can be used to describe an ellipse. Its eigenvectors are considered to

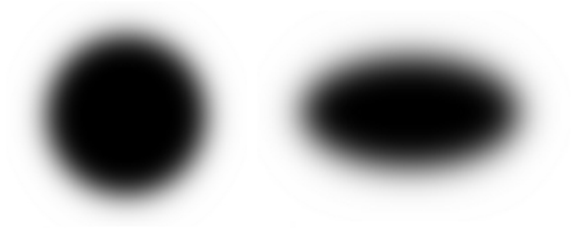


Figure 10.2. Examples of isotropic and anisotropic diffusion

be the direction of the ellipse's two axes and its eigenvalues to be the length of the two axes.

$$A = \begin{bmatrix} A_{xx} & A_{yx} \\ A_{xy} & A_{yy} \end{bmatrix} \tag{10.5}$$

For example, the ellipse presented in Figure 10.3 can be described by three values,$(2, 1, \pi/4)$. The ellipse's axes' lengths are 2 and 1, and the ellipse is rotated 45 degrees ($\pi/4$). Alternatively, it can be described by the matrix

$$A = \begin{bmatrix} \frac{3}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{3}{2} \end{bmatrix} \tag{10.6}$$

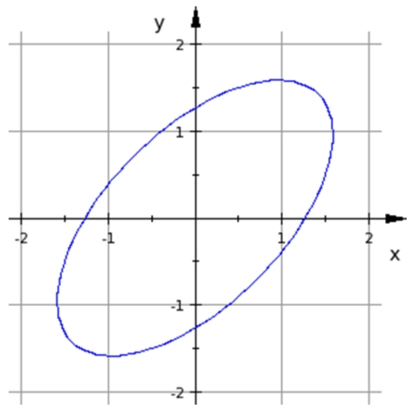


Figure 10.3. Sample ellipse to be described as a matrix

because its eigenvectors are $[-\sqrt{2}/2, \sqrt{2}/2]$ and $[\sqrt{2}/2, \sqrt{2}/2]$ and its eigenvalues are 1 and 2, respectively. We note that the eigenvectors' directions are the same as the ellipse's axes and that the eigenvalues are the length of the axes. Given matrix A, it is also possible to find the ellipse equation.

An ellipse can be used to describe how water spreads on a plain surface, but it is not able to describe how water diffuses inside the brain, because this is a 3D process. Instead, ellipsoids can be used, which can be seen as a 3D version of ellipses. An ellipsoid needs six parameters to be fully specified. They can be summarized in a 3×3 symmetric matrix whose eigenvectors lie along the ellipsoid principal axes and whose eigenvalues are the length of the axes.

The diffusion coefficient (amount of diffusion) along any direction can be computed using the following formula:

$$c(d) = d^T A d = dd^T A \tag{10.7}$$

where, c is the diffusion coefficient in the direction d and A is the matrix describing the diffusion properties of the medium.

It has been noticed that water diffusion, in some tissues, is not equal in all directions, that is, water diffuses mainly in some directions. In this case, a single constant (D) is not enough to describe diffusion fully. It is better to make use of a tensor [160].

As described earlier, solving the Equation (10.7) for all DWI images produces 6 apparent coefficient indexes for a given DTI-scan. Then, it can be combined in a single tensor that describes the preferred diffusion orientation. The reason to use tensors is that they are basis invariant, and a DWI procedure cannot be ensured the basis in which the image is taken, even the lab coordinate system is not valid.

$$\ln\left(\frac{A(b)}{A(b=0)}\right) = -\sum_{i=1}^{3}\sum_{j=1}^{3} b_{ij} D_{ij} \tag{10.8}$$

The final formulation to create a tensor volume based on the DWI series can described by Equation (10.8) that it can be expanded as described in Equation (10.9):

$$-(b_{xx}D_{xx} + 2b_{xy}D_{xy} + b_{yy}D_{yy} + 2b_{yz}D_{yz} + b_{zz}D_{zz}) = -trace(bD) \tag{10.9}$$

where $A(b)$ is echo magnitude of the diffusion weighted, $A(b = 0)$ the echo magnitude of the non diffusion weighted and b_{ij} is a component of the symmetric b-matrix.

The final result is a tensor as presented in Equation (10.10). A tensor needs to be computed to all voxels in the series. Note that no realignment of the diffusion direction series is required since the series are obtained in a very narrow timeframe.

However some researchers [26, 88, 122] point out that even by imaging in a short period, noise and artifacts can be produced. Recently, MRI scans capable of DWI protocols are manufactured with embedded procedures to perform such minor corrections.

$$D = \begin{bmatrix} D_{xx} & D_{xy} & D_{xz} \\ D_{yx} & D_{yy} & D_{yz} \\ D_{zx} & D_{zy} & D_{zz} \end{bmatrix} \tag{10.10}$$

Once the tensor volume is available, a number of post-processing tasks is in order. The simplest one is to compute indices such as Fractional Anisotropy (FA), Mean Diffusivity (MD) and so on. Those indices can be used to produce computer generated images showing the anatomy overlaid by the information of those indices.

10.2 Visualization Schemes

The visualization schemes are usually generated using the encoded information for all diffusion directions of a DWI series. This is due to the fact that differently from other magnetic resonance protocols DWI produces various sets of images instead of only one. Figure 10.4 exemplify four different visualization schemes based on computed indexes for the same section of a DTI volume. In (A) the fractional anisotropy map; (B) apparent diffusivity coefficient; (C) volume ratio; and (D) color map based on the principal diffusion direction. For instance, color maps of fractional anisotropy can be used to aid identification of hippocampus atrophy [119] and in studies referring to multiple sclerosis [24].

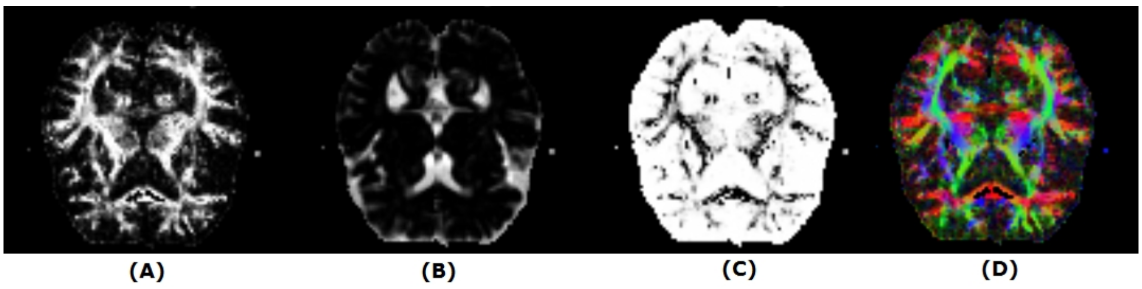


Figure 10.4. Visualization schemes based on computed indexes for DTI

Once the diffusion tensor is available, it is possible to compute a number of different invariant indexes. Those indexes are scalars with invariant values since they hold the same value independent of the tensor orientation. Their real utility is the ability to inform about each point in the voxel's volume using simple scalar values. Basser *et al.* [11] proposed three simple indexes computed using the tensors as basis. They are:

$$I_1 = \lambda_1 + \lambda_2 + \lambda_3 \tag{10.11}$$

$$I_2 = \lambda_1 \cdot \lambda_2 + \lambda_3 \cdot \lambda_1 + \lambda_2 \cdot \lambda_3 \tag{10.12}$$

$$I_3 = \lambda_1 \cdot \lambda_2 \cdot \lambda_3 \tag{10.13}$$

in which λ_1, λ_2, and λ_3 are the eigenvalues of the tensor. The invariance property of those indexes derive from the fact that they are based on the tensor's eigenvalues that are themselves invariant.

Based on those three very simple indexes a number of representative concepts were derived and used directly or indirectly by DTI visualization schemes. Perhaps the most common of those indexes is called Mean Diffusivity (MD) also commonly denoted $\langle D\rangle$ in the pertinent literature. It is computed by averaging the three eigenvalues[2] as follows:

$$MD(D) = \langle D\rangle = \frac{\lambda_1 + \lambda_2 + \lambda_3}{3} \tag{10.14}$$

Visualization schemes based on MD alone are not necessarily an improvement compared to the raw plotting of DWI images. The MD index makes no differentiation between the isotropic and anisotropic parts of the tensors. Basser *et al.* [12] proposed the decomposition of diffusion tensors in theirs two components (iso and anisotropic) and based on this decomposition, two new indexes, namely fractional anisotropy and relative anisotropy were proposed. The tensor decomposition can be achieved by means of the following equation:

$$D = \langle D\rangle I + (D - \langle D\rangle I) \tag{10.15}$$

in which I corresponds to the identity tensor, the first parcel (Equation (10.16)) of the sum to the isotropic and the second (Equation (10.17)) to the anisotropic parts of the tensor. For sake of clarity they can be rewritten as:

$$D^i = \langle D\rangle \tag{10.16}$$

$$D^a = D - \langle D\rangle I \tag{10.17}$$

[2]The sum of the three eigenvalues is also commonly referred as the trace of the tensor $tr(D) = \lambda_1 + \lambda_2 + \lambda_3 = d_{xx} + d_{yy} + d_{zz}$

The final concept necessary to properly define the FA and RA indexes is the magnitude of a tensor, which is essentially the square product of the tensor. It can be computed as follows:

$$|D| = \sqrt{D \cdot D} = \sum_{i=1}^{3} \sum_{j=1}^{3} d_{ij}^2 = \lambda_1^2 + \lambda_2^2 + \lambda_3^2 \tag{10.18}$$

Based on this formulation, FA is define as the proportion of the anisotropic component by the magnitude of D. FA index assumes values within the range $[0, 1]$ in which $FA \approx 1$ is produced if one of the diffusion directions is conceivably large than the others ($\lambda_1 \gg \lambda_2 \approx \lambda_3$) and 0 if the median is totally isotropic. It can defined as described in Equation (10.19).

$$FA(D) = \frac{\sqrt{3}}{\sqrt{2}} \cdot \frac{\sqrt{D^a \cdot D^a}}{\sqrt{D \cdot D}} \tag{10.19}$$

Relative anisotropy is define as the proportion between the anisotropic and isotropic parts of the tensor. It is defined as follows:

$$RA(D) = \frac{\sqrt{D^a \cdot D^a}}{\sqrt{D^i \cdot D^i}} \tag{10.20}$$

By computing those indexes for all tensors in the volume, it is possible to plot sections of presenting such scalars. Fractional anisotropy is especially useful since it provides a clear view of the anisotropic component of each tensor. There is still a myriad of indexes based on the tensors proposed in the literature. For example, the Coherence Index (CI) [92] is yet another one measuring the coherence of axons between neighboring voxels. It is done by comparing the directions of the primary eigenvectors. Shimony *et al.* [135] defined among others an index based on the tensor's variance.

Although informative, and in many cases sufficient for diagnostic purposes, visualization schemes based on scalar indexes limit the visualization procedure to plotting of 2D sections of the volume. Perhaps the biggest advantage of the DTI protocol is that it provides the basis for visualization of fibrous tissues such as the brain white matter in a 3D space. But for that, a more complex post-processing step is required. It is called fiber tracking. The main idea is to use the anisotropic diffusion information encoded within the tensors to infer the pathways of fibers. The next section reviews in more details this post-processing step.

10.3 Fiber Tracking

Fiber tracking [28] is a post-processing step applicable to DWI scans. It became very popular since it allows the visualization in three dimensions of pathways of fibers such as muscles or neural tracts within the scanned region. Examples of fiber tracking of the human brain are presented by Figure 10.5 in which stream lines (left) and stream tubes (right) are used to plot the fibers. The significance of this new imaging method is given by the fact that until the advent of DWI and consequently fiber tracking, there was no way except dissection studies to inspect such structures. In fact, e.g. the white matter shows as a homogeneous region in both CT and MR scans (except for the DWI protocol). The same applies to muscles such as the heart.

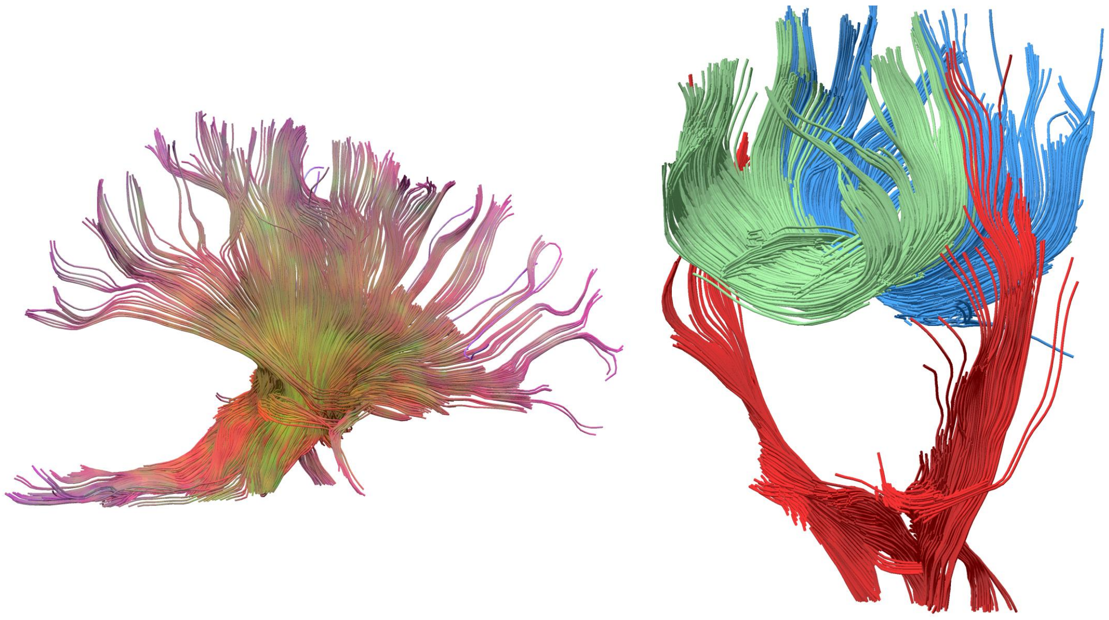


Figure 10.5. High resolution fiber tracking rendering

The general idea behind fiber tracking is that, after computing the tensor's volume, to use the diffusion direction information encoded in each tensor associated to the voxels in the original volume to infer the direction the fibers travels. This is possible because as stated before, in fibrous tissues the water molecules tend to have their motion constrained by the fibrous structure, moving preferably alongside them. Instead of directly imaging the fibers themselves, their "shadow" represented by the water's molecules motion is captured. Since the distance between the actual fiber and the molecules is very small, this method proves to be very effective.

The most common fiber tracking method is based on the idea of line propagation [11]. It is a very intuitive method. As first step, it stipulates that a set of seed points should be defined. They can either correspond to the voxel's centers or to be generated using any desired resolution. If the seed points do not correspond to the voxel's centers, an additional step regarding tensor interpolation is required. The next step defines that for each seed point s_i the corresponding tensor should be computed. Given the tensor t_i its eigenvectors e_i are estimated. The method follows by assuming that the main diffusion direction is given by the eigenvector corresponding to the highest eigenvalue that corresponds to the fiber alignment at the point in the volume's space. The next point s_{i1} is then computed using as reference the current point and the eigenvector ϵ_1 by the following equation:

$$s_{1+1} = s_i + h\epsilon_1 \tag{10.21}$$

in which ϵ_1 is the principal eigenvector and h is an incremental step value.

Figure 10.6 exemplifies the line propagation method. Starting at the seed point x_0, the main diffusion direction is assumed to be the one specified by the eigenvector corresponding to the largest eigenvalue[3] λ_1. Using ϵ_1 as directional component, the next point of the fiber is computed using Equation (10.21) until the stop criteria is reached. Note that for the example given, a 2D representation is presented for sake of simplicity. However, the ellipses represented are in fact ellipsoids. It is also interesting to note that for a single seed point, there are two possible directions to be followed, namely posterior and anterior directions.

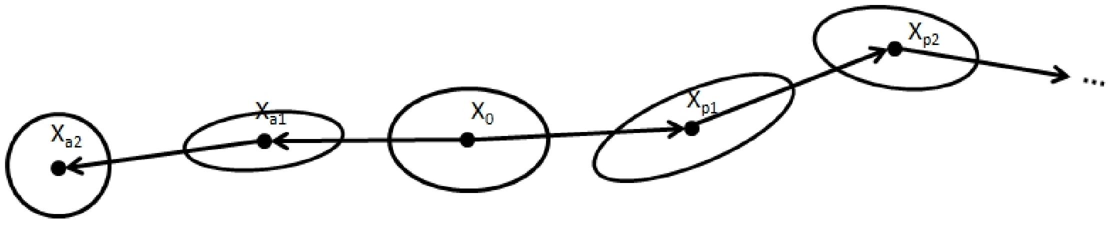


Figure 10.6. Example of line propagation method for one seed point

[3]the eigenvalues are commonly sorted in decreasing order being λ_1 the largest and λ_3 the smallest.

Algorithm 10.7 Fiber tracking algorithm

```
Input: $TV$ tensor volume,
       $h$ incremental step
Output:a fiber tractography

1. Create initial seed points $S$
2. For all seed points in S
3.      Identify the tensor $t_i$ corresponding to $s_i$
4.      $F\_ant \leftarrow$ line_propagate($s_i$, $TV$, $h$, $anterior$)
5.      $F\_pos \leftarrow$ line_propagate($s_i$, $TV$, $h$, $posterior$)
6. End
```

Line propagation algorithms follow both directions until the stop criterion is reached. Regarding the stop criteria, it is important to note that, this is the main difference between the fiber tracking methods based on line propagation concept. The most common criterion refers to a maximum number of points for each fiber. This is usually used in conjunction with other criteria such as when the algorithm reaches a node with low anisotropic diffusion index. Based on this consideration, Westin *et al.* [165, 164] classified the measurable diffusion in three cases, based on the tensor's engenvalues:

- Linear - $\lambda_1 \gg \lambda_2 \approx \lambda_3$ - the diffusion happens mainly in the direction given by ϵ_1;
- Planar - $\lambda_1 \approx \lambda_2 \gg \lambda_3$ - the diffusion happens mainly in the plane defined by ϵ_1 and ϵ_2;
- Spherical - $\lambda_1 \approx \lambda_2 \approx \lambda_3$ - the diffusion is isotropic.

Based on this definition, a plausible stop criterion is given by the line propagation algorithm reaching a note in which the diffusivity is spherical, in other words, isotropic (See Figure 10.6). More sophisticated stop criteria rely on indices such as FA or RA to inform about the confidence of a given node to belong to the fiber being tracked.

Algorithm 10.7 presents a simple version of line propagation. As stated before, the seed points are generated. They are the starting point from which each individual fiber in the tractography is constructed. By selecting the voxel's centers as seed points, the method can be greatly simplified. However, the resolution will be

Algorithm 10.8 Line propagate algorithm

```
Input: $s_i$ seed point,
       $TV$ tensor volume,
       $h$ incremental step,
       $v$ direction of the propagation
Output: a fiber tractography

1. Proceed according to the direction specified in $v$
2. Add $s_i$ to the current fiber being tracked
3. Proceed while stop criteria not reached
4.     Compute the eigenvectors and eigenvalues of the tensor $t_i$
5.     Compute the next point $s_{i+1} = s_i + h\epsilon_1$
6.     Add $s_{i+1}$ to the current fiber being tracked
7.     $s_i \leftarrow s_{i+1}$
8. End
```

considerably poorer if compared with a grid of seed points generated using smaller distances. The drawback of additional seed points is computational time required for the tractography. It is proportional to the number of seed points. Another consideration is the need of tensor interpolation in order to use seed points other than the corresponding to voxel's centers. This is due to the fact the original tensor volume computed based on the DWI images has no corresponding tensor available to that specific point in the continuous space, therefore an interpolation is needed. Additionally, for every point found during the line propagating algorithm, interpolated tensors have to be computed, since they rarely coincide to the original tensor's volume.

The line_propagate procedure is responsible for performing the line propagation as well to ensure the stop criteria defined by the heuristic is reached. Once the fiber is tracked for the two possible directions (anterior and posterior) the resulting sub-fibers are merged producing the final tracked fiber for a specific seed point. Algorithm 10.8 shows the line propagation part.

The line propagation procedure starts by deciding to which direction (anterior or posterior) the fiber should be tracked. This is done by deciding which direction of the principal eigenvector to follow. The next step assigns the seed point as belonging to the current fiber. A loop is started in which the next point is computed. This is done by finding the corresponding tensor to the current point, in order to be able to access their eigenvectors and eigenvalues. The new point is identified based on

Equation 10.21 and subsequently added to the current fiber being tracked. The process continues until the stop criteria are reached.

Fiber tracking algorithms suffer with high signal/noise ratio existing in raw DWI images. It also experience problems with fiber discontinuity. It occurs when the fiber tracking algorithm is unable to decide if the fiber actually ended or which direction it should take. Crossing and kissing of fibers is another important situation any successful fiber tracking algorithm should be able to differentiate. Fiber kissing (Figure 10.7-A) occurs when two fibers pass through a voxel and they continue without actually intersecting each other. Fiber crossing (Figure 10.7-B) as the name suggested and it occurs with the fibers arriving in a voxel crosses. In cases that the algorithm is unable to decide between the two cases, erroneous fibers are potentially generated, invalidating the fiber tracking as a whole. This can culminates in incomplete or erroneous FT results, where phantom curves or outliers are prone to appear. New ways to aid FS methods are continuously proposed to address such problems. In fact several fiber tracking algorithms [11, 20, 28, 120, 138, 147] do exist, implementing different heuristics to address the problems described above.

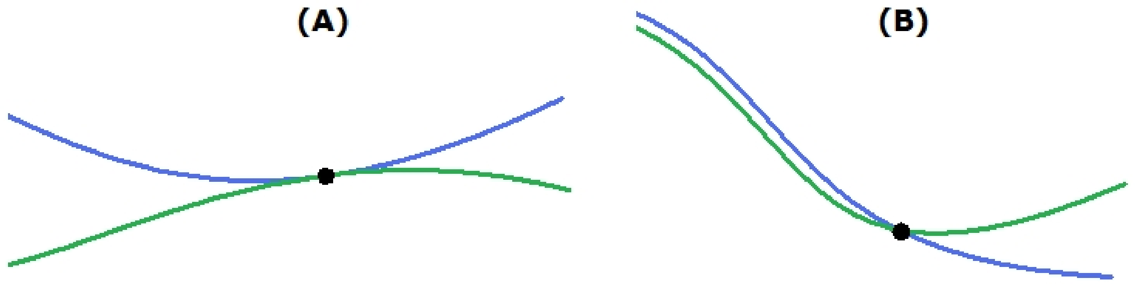


Figure 10.7. Example of fibers kissing and crossing

Although capable of creating a good non-invasive representation of living fibrous tissues such as the human brain's white matter, the output of fiber tracking algorithms can be very confusing for visual inspection. Given the potentially large number of fibers produced as well the 3D geometrical complexity of such maps. During the last years, ways to classify subsets of fibers, mapping them to known anatomical regions are the subject of intensive study. The process of subdivide the whole of fibers into subsets, bundles or tracts is called fiber segmentation.

As stated before, the computational time required by fiber tracking algorithms is proportional to the resolution of the tracking being processed. This is directly translated to the number of seed points used. Recently, Imaging equipments capable of DWI protocol come with embedded software capable of performing fiber tracking. However, it is still the common practice to archive the raw DWI scans. An interesting algorithm based on line propagating method capable of track a whole volume with high resolution was proposed in [118]. The merit of such work relies on the fact it uses GPUs (Graphic Processing Units), commonly available in modern graphic cards, to process the huge amount of computations needed during the tracking of a whole volume. The tractographies presented in Figure 10.5 were produced using this method.

This chapter presented a short review about the MR-DWI protocol and its most common post-processing steps. It presented some historical data regarding this protocol and discussed the merits of it. The underlying mathematics used by the method was reviewed and the most common visualization schemes based on scalar values were presented. The chapter concluded by presenting the general idea in which is based the fiber tracking post-processing procedure. Fiber tracking is the expected input required for any fiber segmentation method. For the purposes of this thesis, it is assumed fiber segmentations are provided. They were generated using the MedINRIA toolset [148] given the fact this tool is well known by the pertinent scientific community, allowing the replication of the results presented.

Chapter 11

Fiber Segmentation

With the introduction of the DWI (Diffusion weighted Images) around two decades ago, a new helm of possibilities concerning the non-invasively investigation of the human brain started. The advantages provided by this new imaging method cannot be directly accessed by inspecting DWI [18] images. DWI itself has little to non clinical value, but the information that can be extrapolated, allows the inspection of the micro-structural anatomy of living tissues. Figure 11.1 shows from left to right, three examples of DWI, MRI-T1 and MRI-T2 images respectively. As it can be seen, it provides poorer anatomical quality if compared to other MRI (Magnetic Resonance Imaging) protocols such as T1 or T2. In fact, DWI is not designed to be an anatomical series, but as a mean to inspect the inner structure of the brain white matter. Any successful usage of DWI series requires some kind of post-processing. Most post-processing methods are related to visualization issues. In fact, one of the first visualization methods attempted was the computation of fractional anisotropy [12]. Color maps [49] based on the principal diffusion directions are also extensively used to improve DTI visualization.

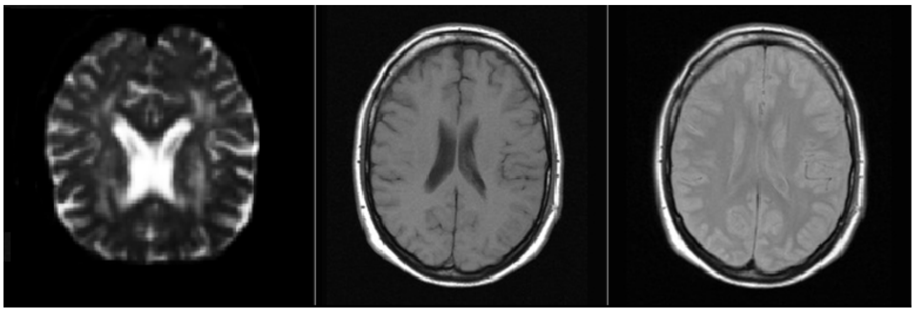


Figure 11.1. Examples of DWI, T1 and T2 MRI images

The most popular post-processing of DWI resides elsewhere. The tensor volumes can be further processed in order to generate detailed mappings of the white matter fiber connectivity of the brain [126], a process called fiber tracking (FT). FT [11] gives access to a new kind of information. First, they show a very detailed representation of the white matter, allowing differentiation on its inner structures (hard to be seemed in anatomical imaging). The second improvement refers to the fact that individual fibers or bundles of them, also called FB (Fiber Bundles) represent the connection between regions of the brain and/or the nerve system. FTs can be tricky to visualize and interpret.

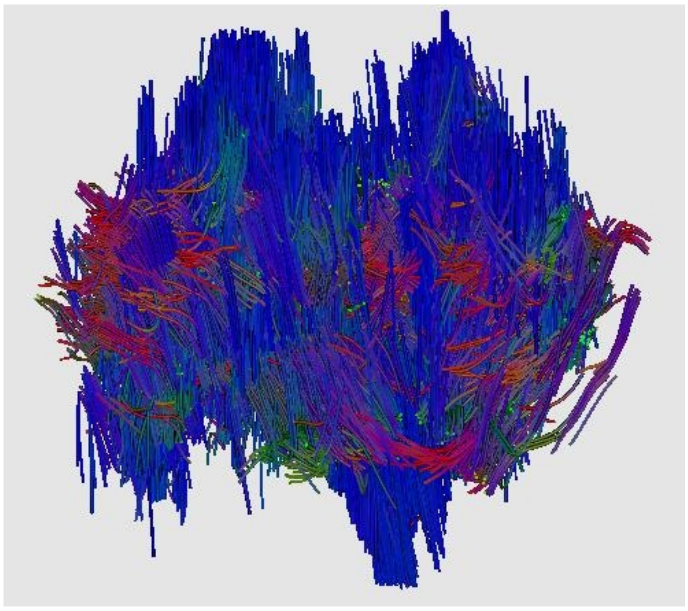


Figure 11.2. Fiber tracking produced with the MedINRIA software

Although presenting a beautiful aspect, as it can be seen in Figure 11.2, raw fiber trackings are hard to inspect. In this particular image, colors are assigned based on the principal diffusion direction. By inspecting a fiber in all its length, it is possible to see the color changing together with the direction followed. Given its three dimensional nature, and the huge amount of information translated in form of fibers (potentially thousands of fibers), it becomes hard to extract only by visual inspection, useful information in order to aid medical diagnosis. A new task called FS (Fiber Segmentation) takes place in order to assign meaning to the potentially incomprehensive set of fibers. There is a number of ways to address the FS problem. Manual segmentation is the most used method.

This chapter proposes an interactive method to ease the manual segmentation process. Two new innovations are introduced. First, the user is presented with a suggestion of segmentation produced by a clustering algorithm. Subsequently, this suggestion can be refined interactively by defining pairwise linkage constraints between fibers. A new clustering is produced, taking into account the newly defined constraints. The process is repeated until the user is satisfied with the segmentation result.

11.1 Fiber Segmentation Methods

The task of easing the interpretation of fiber trackings via assigning meaning to the subsets of fibers is known as fiber segmentation. Figure 11.3 shows the general framework for fiber segmentation based on the original DWI images. As stated before, the tensor volume computation and the fiber tracking based on it are necessary steps. The general pipeline works as follows. Initially a DWI series is captured using a MRI scanner. It is stored and, subsequently, accessed in order to perform the tensor computation. This process translates the set of weighted images into a single tensor volume. Afterward, a fiber tracking algorithm such as the ones proposed e.g. in [11, 28, 138] is used to create a set of fibers. The fibers represent the interconnections of the white matter, usually not visible in common scanning protocols. The set of fibers is finally processed by means of a fiber segmentation method.

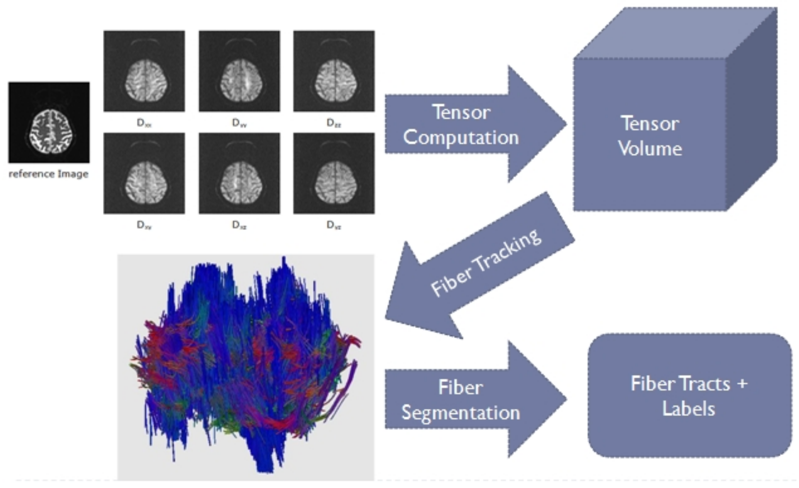


Figure 11.3. General framework for fiber segmentation based on the original DWI images

This process outputs the subsets of meaningful fibers or fiber bundles. They represent the known structures connecting functional regions of the brain. The process is intended to make easier the interpretation of the data, usually hard to inspect due its three-dimensional complexity. Another factor complicating the interpretation is the possibly large number of fibers. It is important to keep in mind that the object of interest in this chapter is the fiber segmentation step.

There are three main research lines dealing with fiber segmentation: a) Interactive; b) Clustering; and c) Atlas-Based. This section describes them in details.

11.1.1 Interactive Segmentation

The main idea of interactive fiber segmentation is to provide visual tools to allow the manual selection of subsets of fibers. Fiber dissection is successfully used to generate comprehensive 3D atlases of the white matter's structure as shown in [159]. The most basic tool regarding manual segmentation is the definition of ROIs (Regions of Interest) [28]. In this process, geometrical shapes are defined by the user. These regions are used to select only the fibers passing through it. Additional ROIs can be specified allowing the segmentation to proceed. A good example of a tool allowing interactive segmentation is the DTI track of the MedINRIA toolset [148] (see Figure 11.4). Although the most commonly used method, the interactive fiber segmentation requires extensive knowledge about the white matters three-dimensional structure. This fact alone motivates the development of quicker and less demanding fiber segmentation solutions.

11.1.2 Clustering Segmentation

Clustering based segmentation methods use cluster algorithms to group fibers into clusters. The grouping is generally based on fiber similarity. This approach is motivated on the idea that fiber sharing similar paths and, with the beginning and the end points possibly close, should belong to the same structure. In order to accomplish it, a suitable distance between fibers is required. In fact, a number of different similarity measures are proposed in the literature. Ding *et al.* [40] proposes a descriptor based on the mean Euclidean distance between pair of points of two curves and the ratio of their lengths. In [29] a fiber descriptor is proposed based on the re-parametrization of each fiber as a Frénet space curve in order to compute the normal, curvature and torsion of each curve in different points, allowing accurate representation of the fibers shape. This method succeeds in describing

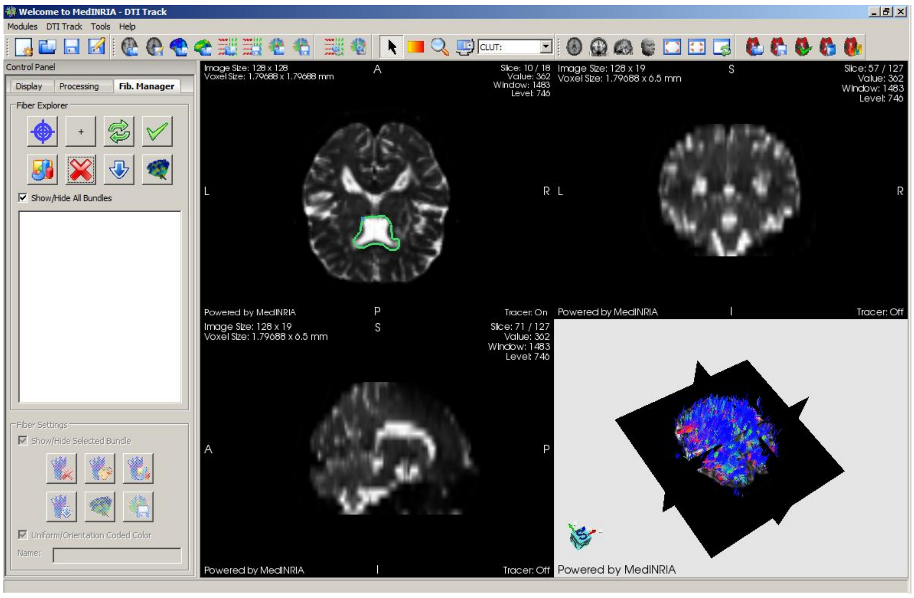

Welcome to MedINRIA - DTI Track
Modules DTI Track Tools Help
Control Panel
Display Processing Fib. Manager
Fiber Explorer
Show/Hide All Bundles
Fiber Settings
Show/Hide Selected Bundle
Uniform/Orientation Coded Color
Name:
Powered by MedINRIA

Figure 11.4. An example of commonly used software for manual fiber segmentation

well the shape of the curves, but it fails in addressing its location. The Euclidean distance is computed between shape descriptors. Finally, in [62] mean closest point distance - MCPD is proposed. This method is by far the most used in the literature. In [29, 62, 176] hierarchical clustering is used to produce the segmentation. Spectral clustering methods are also popular choices as shown in [83] and [124].

Automatic methods, although less demanding from the user point of view, imposes additional complexity in defining a suitable similarity measure between curves able to account to all possible nuances introduced by the complex inner structure of each subset of fibers.

11.1.3 Atlas Based Segmentation

Atlas based segmentation relies on the idea of mapping a FT to a standard space (e.g. stereotaxical space) in which it can be properly correlated with structures or regions. This approach is divided in two parts: a) atlas creation; and b) fiber segmentation. Maddah *et al.* [111] created an atlas via manual/interactive methods. It is also possible to automate at some degree the atlas creation task. O'Donnell [125] proposed a cluster based method to generate an initial segmentation of various subjects. Subsequently, a human interaction step is taken in order to assign labels to each of the clusters, correlating them. Once the atlas is available, the FS is performed by correlating the raw FT to the atlas space. This method requires an initial effort in creating the atlas to which all other FTs are mapped in order to producing a FS. This initial manual effort is balanced by easing further segmentations.

Given the fact brain diseases can alter the brain anatomy, in some cases, very drastically, atlas based methods can be rendered ineffective within the context of real medical applications. This is due to the fact that, if the anatomy of a given brain is too dissimilar to the template used by the atlas based method, a matching (registration) can be problematic.

FT algorithms suffer from fiber discontinuity, crossing and kissing of fibers as well to high noise ratio existing in raw DWI images. Those factors culminate in incomplete or erroneous FT results, where phantom curves and outliers are prone to appear. New ways to aid FS methods find room to be proposed to address such problems. The fiber segmentation method proposed in this chapter can be deemed as a hybrid approach. It is essentially a cluster segmentation with an interactive component. The hybrid nature of the method will become clear in the following sections.

11.2 Fiber Segmentation Using Constrained Clustering

Automatic methods are less demanding from the user point of view, but they impose additional complexity in defining a suitable similarity measure between curves. It is required that, similarity measures to able to account to all possible nuances introduced by the complex inner structure of each real subset of fibers. Given the fact brain diseases can alter the brain anatomy, in some cases, very drastically, atlas based methods can be rendered ineffective within the context of real medical applications.

Usually, medical institutions with access to fiber tracking technology, using it in a daily clinical basis, rely on manual segmentation. The reasons driving this choice of tool can be summarized as follows:

- The user has the complete control of the segmenting process;
- It is easier to account to anatomical changes induced by the disease being studied, resulting from chronic diseases, natural malformations, or anatomical abnormalities;

- The user tends to feel more confident in relying on a result of which he has the complete control over the process of obtaining it.

With these reasons in mind, a new fiber clustering method related to the model in [29] is therefore proposed. In this approach, incorrect tracts splitting and merging are addressed via the definition of must-link and cannot-link constraints. The method is depicted in Figure 11.5. Initially, a FT is produced generating a set F of N fibers $f_i = \{x_i, y_i, z_i\}$ individually represented by an ordered list of 3D points. Each fiber can, potentially, contain different number N of points.

The raw fiber segmentation is presented using a 3D visualization tool, allowing the user to define must-link and cannot-link constraints between key fibers or clusters. A certain degree of knowledge about the white matter's anatomical structure is advised. Eventually, it is also possible to use this method without the definition of any constraints during the first execution. Constraints can also be defined interactively over an initial segmentation. The next step refers to the computation of a similarity matrix via a pairwise distance between curves. A threshold value T must be specified. This value controls the sensibility with which the fibers are clustered together. Algorithm 11.9 is executed producing an initial segmentation. The partial segmentation result is subsequently, presented to the user. At this point, new constraints can be defined or old ones, removed, sensibility value adjusted and a new clustering is produced. The process continues until the user is satisfied with the FS achieved.

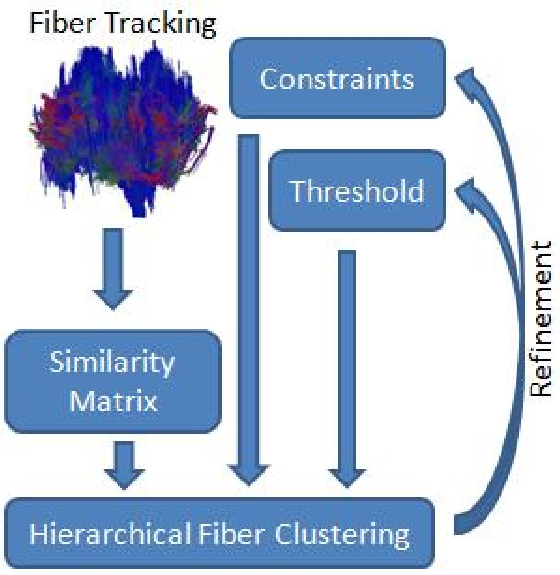


Figure 11.5. General steps of the proposed fiber segmentation method

Additionally, in order to deal with outliers, it is possible to define a minimum acceptance value, in which any produced cluster is regarded as a valid cluster if and only if it contains at least a certain number of fibers. Any fibers filtered by this strategy are presented to the user as not clustered.

11.2.1 Computing Similarity Between Fibers

As discussed in Section 11.1, there is a number of ways to define the similarity between two fibers. In the presented experiments, the mean closest point distance [29] is the option of choice. This is due to the fact that this measure encompasses information about both, shape and location, into a single measurement. It is defined as follows:

$$d_{MCPD}(F_1, F_2) = \frac{1}{n^1} \sum_{i=1}^{n^1} \min_{f_j \in F_2} \|f_j - f_i\| \tag{11.1}$$

where $\|\cdot\|$ is the Euclidean norm; and n^1 is the number of points in F_1.

It is computed by averaging the minimum distance between each point of a reference fiber to the closest point into a second fiber. It is important to point out that MCPD is not symmetric as it can be concluded by inspecting the example given in Figure 11.6. The arrows indicate the direct closest distance from the individual points of fiber i to the closest point in fiber j. There is no guarantee ensuring $d_{MCPD}(f_i, f_j) = d_{MCPD}(f_j, f_i)$. A way to circumvent such difficulty, is to take the minimum of the two possible distance values as shown in Equation (11.2).

$$sim_{MCPD}(F_1, F_2) = \min(d_{MCPD}(F_1, F_2), d_{MCPD}(F_2, F_1)) \tag{11.2}$$

Based on Equation (11.2) an $N \times N$ similarity matrix is computed. It is used as similarity base for the hierarchical constrained clustering algorithm.

11.2.2 Constraints Assignment

The definition of constraints is proposed as a way to perform the merge or separation of fiber and/or tracts that otherwise would be clustered erroneously. There is a number of situations which such cases would occur. Figure 11.7 presents two examples in that the definition of such constraints is helpful. On the left, a must-link constraint is specified between fibers of the cortico-spinal tract in order to ensure

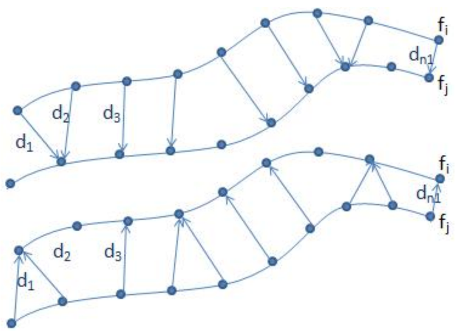


Figure 11.6. Example showing the computation of the MCP distance between fibers f_i and f_j

those two sub-tracts to be merged as a single one. On the right side, a cannot-link constraint is specified in order to ensure separation between the cortico-spinal fibers and the colossal fibers. It is possible to specify whole subsets of fibers to be constrained by manually defining regions. However, this is not necessary, since the definition of a single $\mathcal{ML}$ constraint between two fibers of the sub-clusters suffices to merge them.

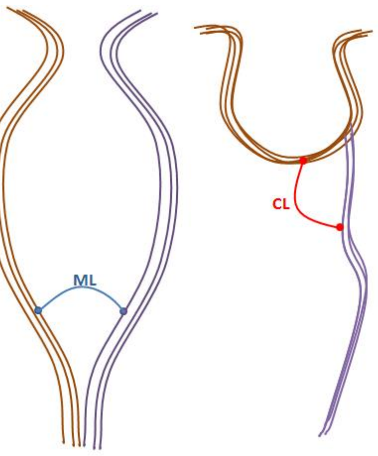


Figure 11.7. Example of fiber constraint specification over toy datasets

Cannot-link constraints ensure that none of the fibers clustered together with a $\mathcal{CL}$ constrained fiber are merged with its counterpart. This kind of constraint is used to fine tuning in cases which specific fibers are not managed by the outlier filter. Figure 11.10 shows a real example of the spinal cord is divided in two clusters and, furthermore, presenting some outliers. In (A) and (B) the sub-clusters present some outliers. Cannot-link are defined between the main bundles and the outliers in order to remove them, in a subsequent iteration. In (C) it can be seen the sub-clusters already without the outliers and finally in (D), they are properly merged to represent the spinal cord.

It is also important to check the transitive closure of $\mathcal{ML}$ and $\mathcal{CL}$ sets of constraints to ensure the triangle inequality property is not violated. Consider the following two must-link constraints, $ml(f_1, f_2)$ and $ml(f_2, f_3)$. These constraints are interpreted as f_1 is must-linked to f_2 and f_2 to f_3. Consider now that no constraint must be violated. This fact implies fibers f_1 and f_3 must be collocated into the same cluster as well, consequently a new constraint is inferred, $ml(f_1, f_3)$. Similarly, consider the $ml(f_1, f_2)$ and $cl(f_2, f_3)$. It is easy to see that f_1 and f_3 cannot be put into the same cluster otherwise the $cl(f_2, f_3)$ constraint is violated. The computation of the transitive closure is very important, in order to speed up the clustering process such as to ensure no deadlocks are encountered, given an invalid classification done in an early stage of the clustering process.

11.2.3 Constrained Fiber Clustering

The pseudo-code presented in Algorithm 11.9 details the envisioned constrained clustering method. In fact, it is an adaptation of the agglomerative clustering proposed in [29]. The algorithm receives as input the set of fibers FT, two sets of constraints, $\mathcal{ML}$ and $\mathcal{CL}$ and a minimum similarity threshold value T.

In step 1, the transitive closure of the $\mathcal{ML}$ and $\mathcal{CL}$ sets is computed, inferring new constraints when needed. This step is important in order to ensure the algorithm will not come to a deadlock. The next step creates of a list of cluster representatives. Cluster representatives are fibers that can potentially evolve into clusters. Initially they are set to be the element appearing in $\mathcal{CL}$ since fibers cannot-link constrained are expected to be in different clusters. Step 2 selects the next cluster representative among the ones not yet assigned to any cluster. In the next step, all available fibers are visited, and if no constraint is violated (condition checked by the $CONSTR\_VIO$ procedure, see [155] for a detailed discussion) it proceeds by checking if the actual fiber can be assigned to the cluster representative.

11.2.4 Threshold Definition and Outlier Detection

The proposed algorithm requires a single parameter in order to define the minimum similarity between fibers. The threshold T ultimately controls the number of final clusters. However, it does not guarantee that a minimum number of fibers will be assigned to each cluster. A small T value would produce many clusters, since only very similar fibers would be clustered together. On the other hand, a large

Algorithm 11.9 Constrained fiber clustering algorithm

```
Input: $FT$ : Fiber tracking
       $\mathcal{ML}$: Set of must-link constraints
       $\mathcal{CL}$: Set of cannot-link constraints
       $T$: threshold
Output: $N$ subsets $C$ of fibers

1. compute the transitive closure over $\mathcal{ML}$ and $\mathcal{CL}$
2. Create list of cluster representatives
3. Select next cluster representatives
4. For all fibers in $FT$
5.     if $CONSTR\_VIO(CC$, current fiber, $\mathcal{CL}, \mathcal{ML})$
6.         Goto 4
7.     end
8.     if any fiber in $CC$ has distance $< T$ compared to the current fiber
9.         assign current fiber to $CC$
10.    end
11.    if any fiber in the last produced cluster occurs in list of cluster representatives
12.    remove such fiber from cluster representatives
13.    end
14. end
```

value would potentially lead to clusters comprising of different anatomical structures. Experimentally, $T = 20$ seems to be a reliable choice for most FTs inspected. This value provides a good trend between initial number of clusters and amount of outliers.

Since the proposed method is interactive, in cases which the user identify that two structures are erroneously merged by the algorithm, it is simple to control the value of T and deal with the sub-structures potentially produced by introducing $\mathcal{ML}$ constraints. Additionally, the user can define a minimum size acceptance level. This value control which clusters are accepted as potentially valid. This is useful in order to restrict the number of presented clusters once small clusters are most likely composed by outliers or erroneously clustered fibers. The relationship between threshold values and the number of identified clusters can be seen in Figure 11.8. In both graphics, N.Clust represents the number of produced clusters. Accepted clusters refer to the number of clusters accepted after outlier removal. In (A), the outlier removal is set to 2% of the initial size of the fibers set and in (B) to 0.5%. These graphics also show the relationship between clusters found and the number of accepted clusters for different acceptance cutoffs. Any fiber not belonging to the accepted clusters is presented to the user as an un-clustered fiber, allowing the definition of constraints in order to deal with them in subsequent interactions.

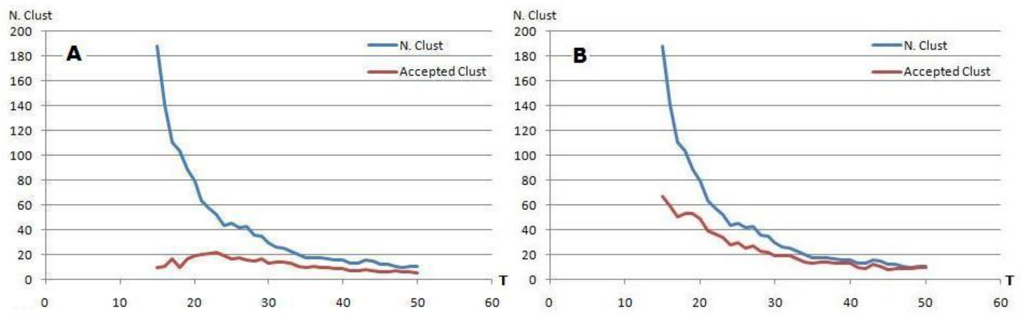


Figure 11.8. Threshold values ranging from 5 to 50 *vs* number of clusters

11.3 Experimental Results

In order to assess the feasibility of the proposed method, it is demonstrated the effective segmentation of a set of tracts. An initial clustering such as the ones presented in Figure 11.9. They are produced without the presence of constraints and presented to the user. $\mathcal{ML}$ and $\mathcal{CL}$ constraints are defined manually allowing further refinement of the FT segmentations. Figure 11.9-(A) shows the results of $T = 30$ and cluster minimum size set to 0.5%;(B) is generated with $T = 10$ and cluster minimum size 2.0%.

Figure 11.10 shows the refinement process in order to segment the spinal cord fibers. (A) and (B) shows two clusters produced referring the spinal cord. In (C) the sub-clusters are already shown without the outliers. Figure 11.10-(D) shows the final merged tract. Bellow, the segmented fibers are projected in the coronal and sagital views. Initially the clusters comprising such structure are identified. It can be seen that some outliers are assigned to both sub-clusters. Cannot-link constraints are interactive defined between the main bundle and the outliers until all such curves are removed. Not necessarily all outliers need to be cannot-link constrained since, there is a chance the algorithm will potentially create another sub-cluster with the remaining fibers, in a subsequent interaction.

Once only the main sub-bundles remain, a single must-link constraint is sufficient to merge them. For this example, as pointed before, only 1 must-link constraint and 9 cannot-link constraints are used to perform the segmentation. The final tract is composed of 713 spinal fibers.

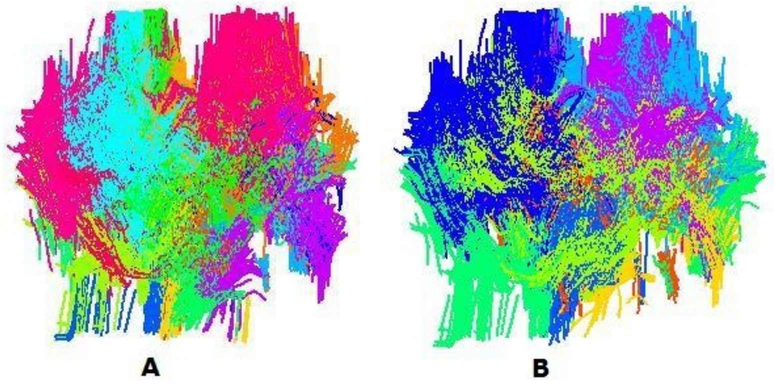


Figure 11.9. Complete plot of initial unconstrained clustering

Another interesting example refers to the segmentation of the corpus collosum. In Figure 11.11 (A), (B), (D) and (E) shows axial and coronal views of an identified cluster. (C) and (F) shows a 3D plot of such fibers. (G) and (H) show the sagital and axial projections of 7 colossal clusters such as the ones presented above into a single cluster. (I) is the 3D plot of the whole tract. This particular region is relatively difficult to segment since it represented the inner core of the brain connecting most of its functional regions. Such complexity is translated as very dissimilar fiber shapes. The clustering algorithm produces a series of sub-clusters containing colossal fibers. They are visually identified by its position in relation to the anatomical reference images. To each sub-clusters, cannot-link constraints are defined in order to remove the subsets of fibers that not belong to this structure. Thus, must-link constraints are defined to finally segment the whole structure in a subsequent iteration.

This chapter proposed to use pairwise linkage constraints to aid the process of fiber clustering. A constrained hierarchical clustering algorithm was developed to accommodate such constraints. It uses as agglomerative hierarchical clustering strategy guided by a threshold parameter to drive the merge of similar fibers into clusters. This method allows automatically to decide the number of clusters, it is directly dependent of the degree of threshold similarity set by the user. The MCP distance was used as similarity measure. The Constraints are defined over the initial segmentation allowing interactive refining of tracts by merging clusters or removing sets of fibers. The usage of pairwise constraints to refine the segmentation seems to be more intuitive than the traditional method comprised by the definition of ROIs, since it only requires the specification of likewise. It also provides a simple way to deal with the problem of outliers, which can be removed by simply defining a cannot-link constraint between them and the main tract. The feasibility of this method is shown through the segmentation of the spinal and colossal fibers. The last

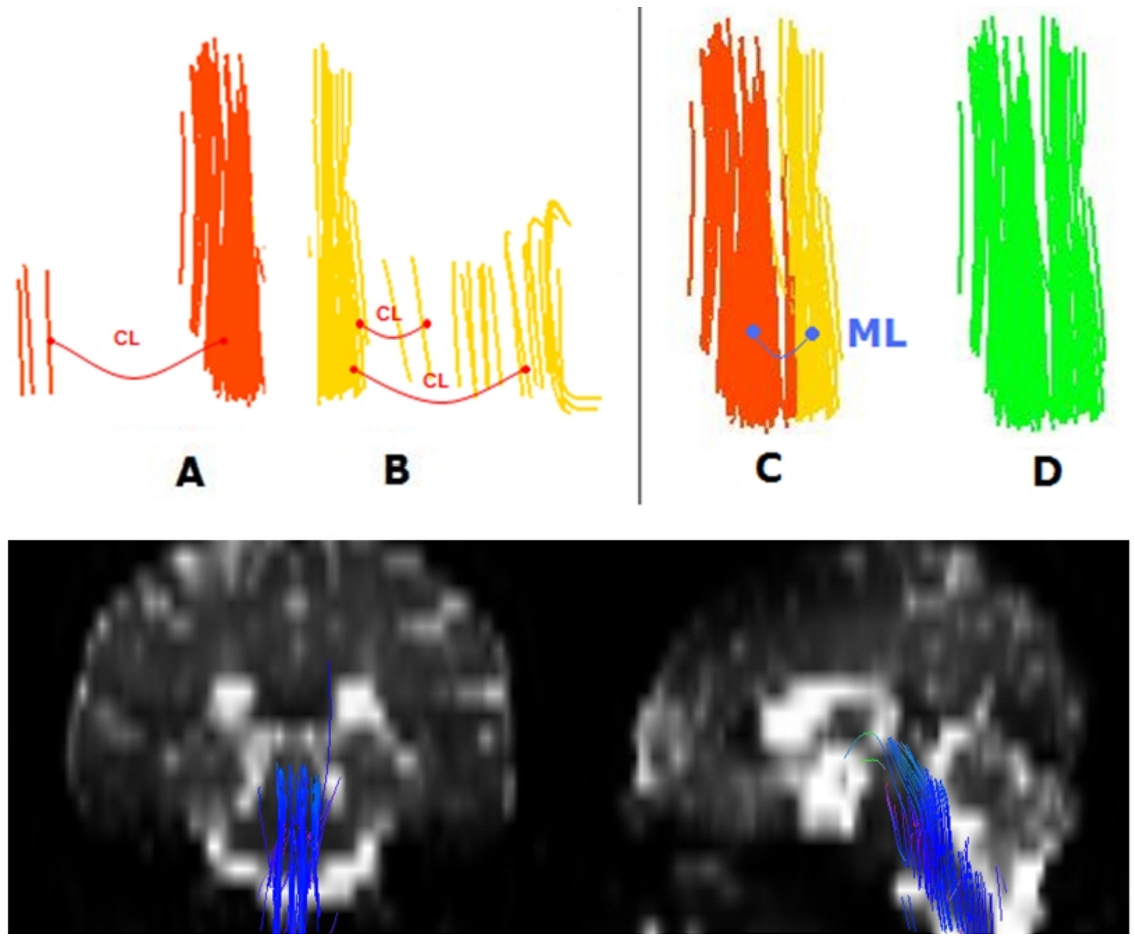


Figure 11.10. Constraints definition for the spinal cord segmentation

example is especially interesting since it poses a major difficulty given the complex special distribution of the fibers belonging to it.

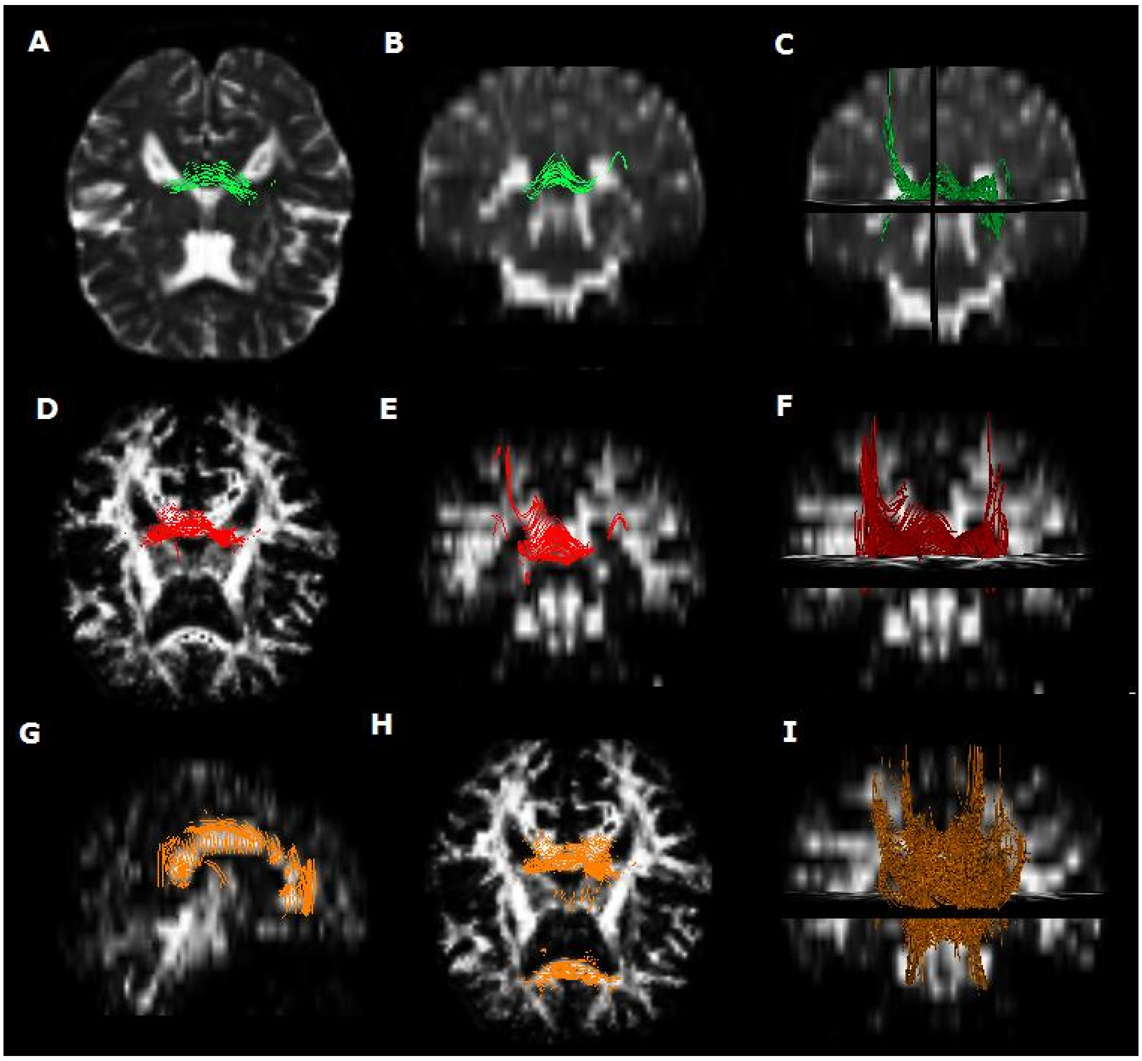


Figure 11.11. Iterative segmentation of colossal fibers

Chapter 12

Conclusion

This thesis presented further developments in the areas of ensemble and constrained clustering. New advances in the field of ensemble clustering were made. The importance of a consistent ensemble generation method was investigated and the central role played by ensemble variability was identified. Therefore, new methods for measurement and visualization of ensemble variability were proposed. A new lower bound for ensemble clustering was an important contribution made by this thesis. The application in general clustering problems of a consensus function based on the random walker formulation was also a topic of interest visited in this thesis. A study was presented evaluating the feasibility of the application of general ensemble clustering methods to the problem of image segmentation combination. Motivated by the ensemble variability study, a new consensus function was introduced based on the sum of pairwise distances formulation. The methods of constrained and ensemble clustering were combined in order to create a new clustering method considering both altogether. Finally, a semi-supervised fiber segmentation framework was proposed using some of the methods investigated in this thesis.

To summarize, the main contributions of this thesis work, in order of appearance in the text, are:

- *Ensemble variability assessment.* A series on contributions were made in this field. First, various ensemble generation strategies were proposed motivated by the fact that many works related to ensemble clustering commonly use very simple generation methods. Subsequently an index called C_{var} was introduced. Based on the median partition formulation, it is capable of giving a measure of how dissimilar the partitions of an ensemble are. It returns a score within the $[0, 1]$ range, where 0 indicates that all partitions in the ensemble are exactly the same, and 1 indicating a high degree of variability. However, since it is simply a measurement of variability, such index is incapable of differentiating if subsets of partitions present low variability. For that, a visualization scheme based on multidimensional dimension reduction was proposed. Due to the dimension reduction, it allows a simple 2D plotting of partition representatives, in which closer points represent similar partitions. Similarly, points far apart are expected to have a high dissimilarity. Using such visualization scheme, that does not require knowledge of a ground-truth, it is possible to assess if a given ensemble is worth to be processed by an ensemble clustering method. For our knowledge, this is the first attempt in visualizing the variability of ensemble of partitions.

- *Ensemble clustering software.* Due to the multitude of ensemble clustering consensus functions proposed in the past few years, and the relative difficulty found in obtaining them for study purposes, a simple software capable of running various consensus functions was proposed. It has a very compact user interface programmed in Matlab. It requires as input, the original dataset and a file containing the ensemble of partitions. A set of consensus functions were programmed and are already available. Additionally, it is easy to extend it to accommodate new consensus functions by programming a new Matlab function and adding a reference to it in the configuration file. Such software, however, allows simple and quick access to most of the work developed in ensemble clustering up to date.

- *Random walker consensus function for general clustering.* The random walker based consensus function originally proposed within the context of image segmentation combination was adapted to be used in general ensemble clustering problems. The main challenge was regarding the generation of a feasible graph representation of the ensemble, necessary for the simulation of the random walks. A graph generation scheme based on the neighborhood of closest patterns was proposed and it was shown experimentally that a very small neighborhood can be used, since the difference in accuracy compared to larger neighborhoods is negligible. The method was compared with well known consensus functions and, its performance was attested in both accuracy related to a known ground-truth and computational complexity.

- *Lower bound for ensemble clustering.* A lower bound Γ specifically designed for ensemble clustering was proposed to explore the question of how well the consensus functions perform in an absolute sense, i.e. how they compare to the unknown optimal solution or ground-truth. It was shown experimentally that for some cases, in which the results attainable by a given consensus function are closely compared to the lower bound, it is safe to assume a good result was obtained. However, in cases which the distance between the result and the lower bound is larger, any making claims must be carefully considered. It is possible to interpret such cases as situations that the consensus function was unable to achieve a good result. This can occur because a number of factors, such as the ensemble is not representative enough or the consensus function is not suitable for that specific application.

- *Sum of pairwise distances.* Motivated by the ensemble variability study, it was observed that ensembles composed by dissimilar partitions can lead to worst results when processed by many of the consensus functions available. Therefore, a new consensus function was introduced. It was shown experimentally that in most cases it can achieve very good results compared to the state of art consensus functions.

- *Constrained ensemble clustering.* A new clustering method combining features of both constrained and ensemble clustering was proposed. Since any ensemble clustering method is comprised of two steps, namely generation and consensus, the question was raised about the necessity/sufficiency of considering the set of constraints only in the generation, consensus or in both steps. The main advantage is that this new method can be used to address the situations that exist in both approaches. It was proven experimentally that in cases where the constraints are considered only in the generation step, constraint's violations are prone to occur. This means that consensus partitions can be generated in which the set of constraints is not fully satisfied. Furthermore, the accuracy tends to be considerably lower if compared to other methods. It was also observed that if the constraints are considered only during the consensus step, no constraint violations will be observed. However, the performance is in most cases considerably lower if compared to the case in which the constraints are considered in both generation and consensus steps. To our knowledge, this was the first time constrained and ensemble clustering were considered together into a single clustering framework.

- *Fiber segmentation.* A novel approach to fiber segmentation by applying constrained clustering was proposed. It consists of a baseline constrained clustering algorithm that allows a reliable segmentation in an interactive manner. The user only is required to formulate high-level knowledge in form of pairwise relationships. It uses a threshold based clustering algorithm that priorities the formation of clusters among neighbor fibers promoting the automatic decision of number of clusters, directly dependant of the threshold value. The MCPD distance was used as similarity measure between fibers. The feasibility of this method is shown through the segmentation of the spinal and colossal fibers. The popular segmentation methods based on clustering alone will hardly produce high-quality segmentations such as the ones presented, especially in abnormal cases. If compared to ROI-based methods, similar results are possible to be obtained. However, the time and effort required by the user is considerably diminished by the proposed approach.

While the preliminary results are very promising, several issues remain. The envisioned further developments can by summarized as follows:

- *Automatic ensemble generation.* It was observed that the ensemble generation step plays a vital role in the ensemble clustering process. The development of a framework specifically designed to automatically generate ensembles using different ensemble generation techniques is envisioned. This framework will be capable of measuring the variability of an ensemble during the generation process. Based on this measurement, it will decide if a given partition is deemed acceptable of if it be re-generated before it is incorporated to the ensemble of partitions. This is essentially what is done in this thesis, except for the fact the ensembles were inspected by a human evaluator;

- *SoPD applied to other problems.* An extensive evaluation of *SoPD* to address problems other than ensemble clustering is envisioned. In special, the computation of median graphs is a problem of interest. The median graph computation is a very costly task, therefore it is also intended to study the vector space embedding based approaches for median graph computation.;

- *Constrained ensemble clustering using global constraints.* The idea of ensemble constrained clustering introduced in this thesis presents numerous possibilities for future works. In special, some new methods are envisioned in which constraints will have their influence spread around the neighboring patterns. It is believed that by using constraints in a global way, better results can be obtained as the experiments presented in Chapter 8 seen to indicate. Additionally, constrained clustering methods based on global constraints require the specification of less constraints if compared to methods based on local constraints to achieve similar results. Since constraints can be user defined, any practical application would require the number of constraints to be limited;

- *Fiber segmentation using constrained ensemble clustering.* The constrained fiber segmentation algorithm presented in this thesis can also be further refined. It is believed that the application of an ensemble constrained clustering algorithm such as the ones proposed in this thesis can greatly improve the performance of the fiber segmentation process. It is expected that by using constrained ensemble clustering less iteration cycles will be required to segment individual fibers. However, the methods proposed in Chapter 9 do not meet the requirements of real-time fiber segmentation. Faster ways to perform ensemble constrained clustering need to be devised. A promising possibility is the usage of a constrained version of the random walker consensus function that is under development.

Bibliography

[1] D. D. Abdala, P. Wattuya, and X. Jiang, "Ensemble clustering via random walker consensus strategy," in *International Conference on Pattern Recognition*, 2010.

[2] D. D. Abdala, "Uma metodologia para criação de cérebros médios e mensuração da atrofia relative do córtex," Master's thesis, Universidade Federal de Santa Catarina, 2005.

[3] S. Aljahdali and E. A. Zanaty, "Combining multiple segmentation methods for improving the segmentation accuracy," in *Proceedings of the 13th IEEE Symposium on Computers and Communications*, 2008, pp. 649–653.

[4] P. Arbelaez, M. Maire, C. C. Fowlkes, and J. Malik, "From contours to regions: An empirical evaluation," in *CVPR*. IEEE, 2009, pp. 2294–2301.

[5] J. C. Atine, A. Doncescu, and J. Aguilar-martin, "A fuzzy clustering approach for supervision of biological processes by image processing," *EUSFLAT- LFA*, 2005.

[6] S. Auwatanamongkol, "Inexact graph matching using a genetic algorithm for image recognition," *Pattern Recognition Letters*, vol. 28, no. 12, pp. 1428–1437, 2007.

[7] H. G. Ayad and M. S. Kamel, "Cumulative voting consensus method for partitions with variable number of clusters," *IEEE Transactions on Pattern Analysis Machine Intelligence*, vol. 30, no. 1, pp. 160–173, 2008.

[8] N. Bansal, A. Blum, and S. Chawla, "Correlation clustering," in *Proceedings of the 43rd Symposium on Foundations of Computer Science*, 2002.

[9] A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall, "Learning distance functions using equivalence relations," in *In Proceedings of the Twentieth International Conference on Machine Learning*, 2003, pp. 11–18.

[10] J. Barthelemy and B. Leclerc, "The median procedure for partition," in *Partitioning Data Sets.* AMS DIMACS Series in Discrete Mathematics, 1995, pp. 3–34.

[11] P. J. Basser, S. Pajevic, C. Pierpaoli, J. Duda, and A. Aldroubi, "In vivo fiber tractography using DT-MRI data," *Magnetic Resonance in Medicine*, vol. 44, pp. 625–632, 2000.

[12] P. J. Basser and C. Pierpaoli, "Microstructural and physiological features of tissues elucidated by quantitative-diffusion-tensor MRI," *Journal of Magnetic Resonance, Series B*, vol. 111, no. 3, pp. 209 – 219, 1996.

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

[14] S. Basu, M. Bilenko, and R. Mooney, "Active semi-supervision for pairwise constrained clustering," in *4th SIAM International Conference on Data Mining*, no. 4. 4th SIAM International Conference on Data Mining, 2004.

[15] S. Basu, I. Davidson, and K. L. Wagstaff, *Constrained Clustering - Advances in Algorithms, Theory, and Applications*, V. Kumar, Ed. Chapman & Hall / CRC Press, 2009.

[16] A. Ben-Hur, A. Elisseeff, and I. Guyon, "A stability based method for discovering structure in clustered data," *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, pp. 6–17, 2002.

[17] K. Bennett, P. Bradley, and A. Demiriz, "Constrained k-means clustering," Microsoft, Tech. Rep., May 2000.

[18] D. L. Bihan, J.-F. Mangin, C. Poupon, C. A. Clark, S. Pappata, N. Molko, and H. Chabriat, "Diffusion tensor imaging: Concepts and applications," *Journal of Magnetic Resonance Imaging*, vol. 13, pp. 534–546, 2001.

[19] M. Bilenko, S. Basu, and R. J. Mooney, "Integrating constraints and metric learning in semi-supervised clustering," in *ICML '04: Proceedings of the twenty-first international conference on Machine learning.* New York, NY, USA: ACM, 2004, p. 11.

[20] A. B. M. Bjrnemo, "White matter fiber tracking using diffusion tensor MRI," Masters Thesis in Biomedical Engineering, Linkping University, February 2002.

[21] I. Borg and P. J. F. Groenen, *Modern multidimensional scaling: Theory and applications (Springer Series in Statistics)*, 2nd ed. Springer, Berlin, September 2005.

[22] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24(2), pp. 123–140, 1996.

[23] M. Brun, C. Sima, J. Hua, J. Lowey, B. Carroll, E. Suh, and E. R. Dougherty, "Model-based evaluation of clustering validation measures," *Pattern Recognition*, vol. 40, no. 3, pp. 807 – 824, 2007.

[24] M. Cercignani, M. Bozzali, G. Iannucci, G. Comi, and M. Filippi, "Intra-voxel and inter-voxel coherence in patients with multiple sclerosis assessed using diffusion tensor MRI," *Journal of Neurology*, vol. 249, no. 7, pp. 875–883, 2002.

[25] Y. Chang, D. J. Lee, Y. Hong, and J. Archibald, "Unsupervised video shot detection using clustering ensemble with a color global scale-invariant feature transform descriptor," *EURASIP Journal on Image and Video Processing*, vol. 2008, pp. 1–10, 2008.

[26] B. Chen, H. Guo, and A. W. Song, "Correction for direction-dependent distortions in diffusion tensor imaging using matched magnetic field maps," *NeuroImage*, vol. 30, p. 121 129, 2006.

[27] D. Cohn, R. Caruana, and A. Mccallum, "Semi-supervised clustering with user feedback," Pittsburg, Tech. Rep., 2003.

[28] T. E. Conturo, N. F. Lori, T. S. Cull, E. Akbudak, A. Z. Snyder, J. S. Shimony, R. C. McKinstry, H. Burton, and M. E. Raichle, "Tracking neuronal fiber pathways in the living human brain," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 96, no. 18, pp. 10 422–10 427, August 1999.

[29] I. Corouge, S. Gouttard, and G. Gerig, "Towards a shape model of white matter fiber bundles using diffusion tensor MRI," in *IEEE International Symposium on Biomedical Imaging: Nano to Macro*, vol. 1, 2004, pp. 344– 347.

[30] T. Cox and M. Cox., *Multidimensional scaling*, 2nd ed. Boca Raton: Chapman Hall, 2001.

[31] N. Cristianini and J. Shawe-Taylor, *An introduction to support vector machines : and other kernel-based learning methods*, 1st ed. Cambridge University Press, March 2000.

[32] N. Cristianini, J. Shawe-Taylor, and J. Kandola, "Spectral kernal methods for clustering," in *Advances in Neural Information Processing Systems*, vol. 14, 2002.

[33] B. A. Davey and H. A. Priestley, *Introduction to lattices and order*, 2nd, Ed. Cambridge University Press, 2002.

[34] I. Davidson and S. S. Ravi, "Clustering with constraints: Feasibility issues and the k-means algorithm," in *SIAM Data Mining Conference*, 2005.

[35] ——, "Towards efficient and improved hierarchical clustering with instance and cluster level constraints," University of Albany, Tech. Rep., 2005.

[36] ——, "Hierarchical clustering with constraints: Theory and practice," in *Proceedings of the ninth European principles and practice of KDD*, 5970.

[37] I. Davidson, K. Wagstaft, and S. Basu, "Measuring constraint-set utility for partitional clustering algorithms," in *ECML/PKDD 2006*, 2006.

[38] I. Davidson and S. S. Ravi, "Agglomerative hierarchical clustering with constraints: Theoretical and empirical results," *Lecture Notes in Computer Science*, vol. 3721, pp. 59–70, 2005.

[39] E. D. Demaine and N. Immorlica, "Correlation clustering with partial information," in *Proceedings of the 6th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems.* new Jersey: Princeton, August 2003, pp. 1–13, clustering.

[40] Z. Ding, J. C. Gore, and A. W. Anderson, "Classification and quantification of neuronal fiber pathways using diffusion tensor MRI," *Magnetic Resonance in Medicine*, vol. 49, p. 716721, 2003.

[41] C. Domeniconi and M. Al-Razgan, "Weighted cluster ensembles: Methods and analysis," *ACM Transaction on Knowledge Discovering Data*, vol. 2, pp. 17:1–17:40, January 2009.

[42] C. Domeniconi, D. Gunopulos, S. Ma, B. Yan, M. Al-Razgan, and D. Papadopoulos, "Locally adaptive metrics for clustering high dimensional data," *Data Min. Knowl. Discov.*, vol. 14, pp. 63–97, February 2007.

[43] S. Dongen, "Performance criteria for graph clustering and markov cluster experiments," Centre for Mathematics and Computer Science, Amsterdam, The Netherlands, The Netherlands, Tech. Rep., 2000.

[44] J. Duarte, A. Fred, A. Loureno, and F. Duarte, "Cluster ensemble selection using average cluster consistency," in *KDIR 2009: Proceeding of International Conference on Knowledge Discovery and Information Retrieval*, 2009.

[45] ——, "On consensus clustering validation," in *Structural, Syntactic, and Statistical Pattern Recognition*, ser. Lecture Notes in Computer Science, E. Hancock, R. Wilson, T. Windeatt, I. Ulusoy, and F. Escolano, Eds., vol. 6218. Springer Berlin / Heidelberg, 2010, pp. 385–394.

[46] S. Eubank, H. Guclu, V. S. Anil Kumar, M. V. Marathe, A. Srinivasan, Z. Toroczkai, and N. Wang, "Modelling disease outbreaks in realistic urban social networks," *Nature*, vol. 429, no. 6988, pp. 180–184, May 2004.

[47] R. Fagin and L. Stockmeyer, "Relaxing the triangle inequality in pattern matching," *International Journal on Computer Vision*, vol. 28, no. 3, pp. 219–231, 1998.

[48] M. Ferrer Sumsi, "Theory and algorithms on the median graph. application to graph-based classification and clustering," Ph.D. dissertation, Universitat Autònoma de Barcelona, 2008.

[49] A. Field, Y.-C. Wu, and A. Alexander, "Principal diffusion direction in peritumoral fiber tracts: Color map patterns and directional statistics," in *Annals of the New York Academy of Sciences*, vol. 1064, 2005, pp. 193–201.

[50] V. Filkov and S. Skiena, "Integrating microarray data by consensus clustering," *International Journal on Artificial Intelligence Tools*, vol. 13, pp. 863–880, 2004.

[51] B. Fischer and J. M. Buhmann, "Bagging for path-based clustering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, pp. 1411–1415, November 2003.

[52] E. B. Fowlkes and C. L. Mallows, "A method for comparing two hierarchical clusterings," *Journal of the American Statistical Association*, vol. 78, no. 383, pp. 553–569, 1983.

[53] C. Fraley and A. Raftery, "Model-based clustering, discriminant analysis, and density estimation," *Journal of American Statistical Association*, vol. 97, pp. 611–631, June 2002.

[54] L. Franek, D. D. Abdala, S. Vega-Pons, and X. Jiang, "Image segmentation fusion using general ensemble clustering methods," in *Tenth Asian Conference on Computer Vision*, 2010.

[55] A. Frank, "On kuhns hungarian method a tribute from hungary," Research Group on Combinatorial Optimization., Tech. Rep., 2004.

[56] A. Frank and A. Asuncion. (2010) UCI machine learning repository.

[57] A. L. Fred and A. K. Jain, "Combining multiple clusterings using evidence accumulation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 6, pp. 835–850, June 2005.

[58] A. Fred, "Finding consistent clusters in data partitions," in *In Proceedings 3rd International Workshop on Multiple Classifier.* Springer, 2001, pp. 309–318.

[59] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, no. 5814, pp. 1 136 800–976, January 2007.

[60] J. Froment. (2010) Megawave. [Online]. Available: http://megawave.cmla.ens-cachan.fr/index.php

[61] K. Fukunaga and L. Hostetler, "The estimation of the gradient of a density function, with applications in pattern recognition," *IEEE Transactions on Information Theory*, vol. 21, no. 1, pp. 32–40, January 2003.

[62] S. G. G. Gerig and I. Corouge, "Analysis of brain white matter via fiber tract modeling," in *Proceedings of the 26th Annual International Conference of the IEEE EMBS*, 2004.

[63] R. Ghaemi, M. N. Sulaiman, H. Ibrahim, and N. Mustapha, "A survey: cluster ensemble techniques," in *Proc. of World Academy of Science, Engineering and Technology*, vol. 38, 2009, pp. 644–657.

[64] A. Gionis, H. Mannila, and P. Tsapara, "Clustering aggregation," *ACM Trans. on Knowledge Discovery from Data*, vol. 1, 2007.

[65] I. Giotis and V.Guruswami, "Correlation clustering with a fixed number of clusters," *Theory of Computing*, vol. 2, pp. 249–266, 2006.

[66] M. Gluck and J. Corter, "Information, uncertainty, and the utility of categories," in *Proc. of the Seventh Annual Conference of the Cognitive Science Society.* Hillsdale, NJ: Lawrence Erlbaum, 1985, pp. 283–287.

[67] A. Goder and V. Filkov, "Consensus clustering algorithms: Comparison and refinement," *Proceedings of ALENEX*, pp. 109–117, 2008.

[68] L. Grady, "Random walks for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 11, pp. 1768–1783, 2006.

[69] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, "Cluster validity methods: Part I," *ACM SIGMOD Record*, vol. 31, p. 2002, 2002.

[70] ——, "On clustering validation techniques," *Journal of Intelligent Information Systems*, vol. 17, no. 2-3, pp. 107–145, 2001.

[71] J. Heinonen, *Lectures on Analysis on Metric Spaces.* New York: Springer-Verlag, 2001.

[72] A. Hiaoui and S. Wang, "Median graph computation for graph clustering," *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, vol. 10, pp. 47–53, 2006.

[73] A. Hoover, G. Jean-baptiste, X. Jiang, P. J. Flynn, H. Bunke, D. Goldgof, K. Bowyer, D. Eggert, A. Fitzgibbon, and R. Fisher, "An experimental comparison of range image segmentation algorithms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, pp. 673–689, 1996.

[74] J. P. Hornak. (2010) The basics of MRI. [Online]. Available: http://www.cis.rit.edu/htbooks/mri/

[75] H. Hruschka, "Market definition and segmentation using fuzzy clustering methods," *International Journal of Research in Marketing*, vol. 3, no. 2, pp. 117 – 134, 1986.

[76] L. Hubert and P. Arabie, "Comparing partitions," *Journal of Classification*, vol. 2, pp. 193 – 218, 1985.

[77] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," *ACM Computing Surveys*, vol. 31, pp. 264–323, 1999.

[78] X. Jiang, A. Münger, and H. Bunke, "On median graphs: Properties, algorithms, and applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, pp. 1144–1151, 2001.

[79] X. Jiang and H. Bunke, "Optimal lower bound for generalized median problems in metric space," in *Proceedings of the Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition.* London, UK: Springer-Verlag, 2002, pp. 143–152.

[80] X. Jiang, C. Marti, C. Irniger, and H. Bunke, "Distance measures for image segmentation evaluation," *EURASIP Jornal of Applied Signal Processing*, pp. 209–209, 2006.

[81] Y. Jiang, K.-J. Chen, and Z.-H. Zhou, "SOM based image segmentation," in *Lecture Notes in Artificial Intelligence*, Y. Y. G. Wang, Q. Liu and A. Skowron, Eds. Springer, Berlin, 2003, vol. 2639, pp. 640–643.

[82] Y. Jiang and Z.-H. Zhou, "SOM ensemble-based image segmentation," *Neural Processing Letters*, vol. 20, no. 3, pp. 171–178, 2004.

[83] L. Jonasson, P. Hagmann, J. Thiran, and V. Wedeen, "Fiber tracts of high angular resolution diffusion MRI are easily segmented with spectral clustering," in *Proceedings of 13th Annual Meeting ISMRM*, ser. ISCAS. Miami: SPIE, 2005, p. 1310.

[84] S. Kamvar, D. Klein, and C. Manning, "Spectral learning," 2003.

[85] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM Journal of Scientific Computing*, vol. 20, pp. 359–392, 1998.

[86] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar, "Multilevel hypergraph partitioning: application in VLSI domain," in *DAC '97: Proceedings of the 34th Annual Conference on Design Automation.* New York, NY, USA: ACM, 1997, pp. 526–529.

[87] L. Keuchel and D. Kttel, "Efficient combination of probabilistic sampling approximations for robust image segmentation," in *Pattern Recognition*, ser. Lecture Notes in Computer Science, K. Franke, K.-R. Mller, B. Nickolay, and R. Schfer, Eds., vol. 4174. Springer Berlin / Heidelberg, 2006, pp. 41–50.

[88] D.-J. Kim, H.-J. Park, K.-W. Kang, Y.-W. Shin, J.-J. Kim, W.-J. Moon, E.-C. Chung, I. Y. Kim, J. S. Kwon, and S. I. Kim, "How does distortion correction correlate with anisotropic indices? a diffusion tensor imaging study," *Magnetic Resonance Imaging*, vol. 24, p. 13691376, 2006.

[89] S. Kirkpatrick, J. Gelatt, C. D., and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[90] D. Klein, S. D. Kamvar, and C. D. Manning, "From instance-level constraints to space-level constraints: Make the most of prior knowledge in data clustering," in *Proc. 19th Intl. Conf. on Machine Learning (ICML 2002)*, 2002.

[91] J. Kleinberg, "An impossibility theorem for clustering," in *Advances in Neural Information Processing Systems*, S. Becker, S. Thrun, and K. Obermayer, Eds. MIT Press, 2002, pp. 446–453.

[92] T. Klingberg, "Myelination and organization of the frontal white matter in children: a diffusion tensor MRI study," *Neuroreport*, vol. 10, pp. 2817–2821, 1999.

[93] M. Kretzschmar and M. Morris, "Measures of concurrency in networks and the spread of infectious disease," *Mathematical Biosciences*, vol. 133, no. 2, pp. 165 – 195, 1996.

[94] E. B. Krissinel and K. Henrick, "Common subgraph isomorphism detection by backtracking search," *Software-Practice Experience*, vol. 34, no. 6, pp. 591–607, 2004.

[95] R. Kumar, M. Rawat, and A. Vashistha, "Automatic cluster detection for mobile customer relationship management using GA-KNN conjunction approach," *International Journal of Engineering Studies*, vol. 1, p. 1523, 2009.

[96] H. C. M. Law, A. Topchy, and A. K. Jain, "Clustering with soft and group constraints," in *International Workshop on Syntactical and Structural Pattern Recognition and Statistical Pattern Recognition*, 2004.

[97] M. Leone, Sumedha, and M. Weigt, "Clustering by soft-constraint affinity propagation: applications to gene-expression data," *Bioinformatics*, vol. 20, pp. 2708–2715, 2007.

[98] S. S. Levine and R. Kurzban, "Explaining clustering in social networks: towards an evolutionary theory of cascading benefits," *Managerial and Decision Economics*, vol. 27, no. 2-3, pp. 173–187, 2006.

[99] L.Hansen and P. Salamon, "Neural network ensembles," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, 1990, pp. 993–1001.

[100] T. Li and T. M. O. S. Ma, "On combining multiple clusterings: an overview and a new perspective," *Applied Intelligence*, pp. 1–13, 2009, 10.1007/s10489-009-0160-4.

[101] T. Li, C. Ding, and M. I. Jordan, "Solving consensus and semi-supervised clustering problems using nonnegative matrix factorization," in *Proceedings of the Seventh IEEE International Conference on Data Mining.* Washington, DC, USA: IEEE Computer Society, 2007, pp. 577–582.

[102] M. Lipczak and E. Milios, "Agglomerative genetic algorithm for clustering in social networks," in *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, ser. GECCO '09. New York, NY, USA: ACM, 2009, pp. 1243–1250.

[103] D. Lopresti and J. Zhou, "Using consensus sequence voting to correct OCR errors," *Computer Vision and Image Understanding*, vol. 67, pp. 39–47, 1997.

[104] B. Luo, R. C. Wilson, and E. R. Hancock, "Spectral embedding of graphs," *Pattern Recognition*, vol. 36, no. 10, pp. 2213 – 2230, 2003.

[105] H. Luo, F. Jing, and X. Xie, "Combining multiple clusterings using information theory based genetic algorithm," in *Proceedings of International Conference on Computational Intelligence and Security*, 2006, pp. 84–89.

[106] H. luo, F. Jing, and X. Xie, "Solving cluster ensemble problems by correlation's matrix," in *IEEE Conference on Computational Intelligence and Security*, vol. 1, 2006, pp. 84–89.

[107] M. Luo, Y.-F. Ma, and H.-J. Zhang, "A spatial constrained k-means approach to image segmentation," in *Fourth Pacific Rim Conference on Multimedia*, 2003.

[108] X. Ma, W. Wan, and L. Jiao, "Spectral clustering ensemble for image segmentation," in *Proceedings of the Genetic and Evolutionary Computation Conference*, L. X. et al., Ed., 2009, pp. 415–420.

[109] Y. S. Maarek, R. Fagin, I. Z. Ben-Shaul, and D. Pelleg, "Ephemeral document clustering for web applications," IBM, Tech. Rep., 2000.

[110] J. B. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, L. M. L. Cam and J. Neyman, Eds., vol. 1. University of California Press, 1967, pp. 281–297.

[111] M. Maddah, A. Mewes, S. Haker, W. Grimson, and S. Warfield, "Automated atlas-based clustering of white matter fiber tracts from DT-MRI," in *Med Image Computing and Computing Assisted Intervention. MICCAI 2005.* Cambridge, MA 02139, USA: Med Image Comput Comput Assist Interv. MICCAI 2005, 2005.

[112] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proceedings of the International Conference on Computer Vision*, vol. 2, 2001, pp. 416–423.

[113] V. Meas-Yedid, S. Tilie, and J.-C. Olivo-Marin, "Color image segmentation based on markov random field clustering for histological image analysis," *Pattern Recognition, International Conference on*, vol. 1, p. 10796, 2002.

[114] M. Meila, "Comparing clusterings by the variation of information," *Learning Theory and Kernel Machines*, pp. 173–187, 2003.

[115] B. Mirkin, "Reinterpreting the category utility function," *Machine Learning*, vol. 45, no. 2, pp. 219–228, 2001.

[116] B. G. Mirkin, *Mathematical Classification and Clustering.* Kluwer Academic Press, Dordrecht, 1996.

[117] N. Mishra, R. Schreiber, I. Stanton, and R. Tarjan, "Clustering social networks," in *Algorithms and Models for the Web-Graph*, ser. Lecture Notes in Computer Science, A. Bonato and F. Chung, Eds., vol. 4863. Springer, 2007, pp. 56–67.

[118] A. Mittmann, T. H. C. Nobrega, E. Comunello, J. P. O. Pinto, P. R. Dellani, P. STOETER, and A. v. Wangenheim, "Performing real-time interactive fiber tracking," *Journal of Digital Imaging*, pp. 1–13, 2010.

[119] M. Mller, D. Greverus, C. Weibrich, P. Dellani, A. Scheurich, P. Stoeter, and A. Fellgiebel, "Diagnostic utility of hippocampal size and mean diffusivity in amnestic MCI," *Neurobiology of Aging*, vol. 28, pp. 398–403, 2006.

[120] S. Mori and P. C. M. van Zijl, "Fiber tracking: principles and strategies - a technical review," *NMR in Biomedicine*, vol. 15, no. 7-8, pp. 468–480, 2002.

[121] D. Mumford and J. Shah, "Optimal approximations by piecewise smooth functions and associated variational problems," *Communicantions on Pure and Applied Mathematics*, vol. 42, pp. 577 – 685, 1989.

[122] T. Netsch and A. van Muiswinkel, “Quantitative evaluation of image-based distortion correction in diffusion tensor imaging,” in *IEEE Transactions on Medical Imaging*, vol. 23, no. 7, July 2004, pp. 789–798.

[123] H. P. Ng, S. H. Ong, K. W. C. Foong, P. S. Goh, and W. L. Nowinski, “Medical image segmentation using k-means clustering and improved watershed algorithm,” in *Proceedings of the IEEE Southwest Symposium on Image Analysis and Interpretation.* Washington, DC, USA: IEEE Computer Society, 2006, pp. 61–65.

[124] L. O’Donnell and C.-F. Westin, “White matter tract clustering and correspondence in populations,” *Eighth International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 140–147, 2005.

[125] ——, “Automatic tractography segmentation using a high-dimensional white matter atlas,” *IEEE Transactions on Medical Imaging*, vol. 26, pp. 1562 – 1575, 2007.

[126] C. Poupon, J.-F. Mangin, C. Clark, V. Frouin, J. Rgis, D. L. Bihan, and I. Bloch, “Towards inference of human brain connectivity from MR diffusion tensor data,” *Medical Image Analysis*, vol. 5, p. 115, 2001.

[127] K. Punera and J. Ghosh, “consensus-based ensembles of soft clusterings,” *Applied Artificial Intelligence*, vol. 22, pp. 780–810, August 2008.

[128] W. M. Rand, “Objective criteria for the evaluation of clustering methods,” *Journal of American Statistical Association*, vol. 66, no. 846-850, 1971.

[129] X. Ren and J. Malik, “Learning a classification model for segmentation,” in *Proceedings of the Ninth IEEE International Conference on Computer Vision*, vol. 1, no. 10 - 17, 2003.

[130] A. Robles-Kelly and E. R. Hancock, “A riemannian approach to graph embedding,” *Pattern Recognition*, vol. 40, no. 3, pp. 1042–1056, 2007.

[131] V. Roth, M. L. Braun, T. Lange, and J. M. Buhmann, “Stability-based model order selection in clustering with applications to gene expression data,” in *Proceedings of the International Conference on Artificial Neural Networks*, ser. ICANN ’02. London, UK: Springer-Verlag, 2002, pp. 607–612.

[132] K. Rothaus and X. Jiang, “Constrained clustering by a novel graph-based distance transformation,” in *19th International Conference on Pattern Recognition*, Tampa, 2008.

[133] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, December 2000.

[134] B. Scholkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* Cambridge, MA, USA: MIT Press, 2002.

[135] J. S. Shimony, R. C. McKinstry, E. Akbudak, J. A. Aronovitz, A. Z. Snyder, N. F. Lori, T. S. Cull, and T. E. Conturo, "Quantitative diffusion-tensor anisotropy brain mr imaging: Normative human data and anatomic analysis," *Radiology*, vol. 112, pp. 770–784, 1999.

[136] V. Singh, L. Mukherjee, J. Peng, and J. Xu, "Ensemble clustering using semidefinite programming with applications," *Machine Learning*, vol. 79, no. 1-2, pp. 177–200, 2010.

[137] S.Rao, H. Mobahi, A. Y. Yang, S. S., and Y. Ma, "Natural image segmentation with adaptive texture and boundary encoding," in *ACCV*, 2009, pp. 135–146.

[138] P. Staempfli, T. Jaermann, G. Crelier, S. Kollias, A. Valavanis, and P. Boesigera, "Resolving fiber crossing using advanced fast marching tractography based on diffusion tensor imaging," *NeuroImage*, vol. 30, p. 110 120, 2006.

[139] J. Stefanowski and D. Weiss, "Carrot 2 and language properties in web search results clustering," in *In Proceedings of the First International Atlantic Web Intelligence Conference.* Springer, 2003, pp. 240–249.

[140] A. Strehl, J. Ghosh, and C. Cardie, "Cluster ensembles - a knowledge reuse framework for combining multiple partitions," *Journal of Machine Learning Research*, vol. 3, pp. 583–617, 2002.

[141] A. Strehl, J. Ghosh, and R. Mooney, "Impact of similarity measures on webpage clustering," in *Proceedings of the 17th National Conference on Artificial Intelligence: Workshop of Artificial Intelligence for Web Search.* Austin, Texas, USA: AAAI, July 2000, pp. 58–64.

[142] Z. Su, Q. Yang, and H. Zhang, "Correlation-based document clustering using web logs," in *In Proceedings of the 34th Hawaii International Conference On System Sciences.* IEEE Computer Society, 2001, pp. 3–6.

[143] J. Suri, S. Setarehdan, and S. Singh, *Advanced algorithmic approaches to medical image segmentation: State-of-the-art applications in cardiology, neurology,*

mammography and pathology. Spring-Verlag, London/Berlin/Heidelberg., 2002.

[144] A. Topchy, A. K. Jain, and W. Punch, “A mixture model for clustering ensembles,” in *SIAM Int. Conference on Data Mining*, 2004, pp. 644–318.

[145] A. P. Topchy, A. K. Jain, and W. F. Punch, “Clustering ensembles: Models of consensus and weak partitions,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 12, pp. 1866–1881, 2005.

[146] W. S. Torgerson, “The first major MDS breakthrough,” *Psychometrika*, vol. 17, pp. 401–419, 1952.

[147] J.-D. Tournier, F. Calamante, D. G. Gadian, and A. Connelly, “Diffusion-weighted magnetic resonance imaging fibre tracking using a front evolution algorithm,” *NeuroImage*, vol. 20, p. 276288, 2003.

[148] N. Toussaint, J.-C. Souplet, and P. Fillard, “Medinria: DT-MRI processing and visualization software,” in *Proc. of MICCAI’07 Workshop on Interaction in medical image analysis and visualization*, 2007.

[149] K. Tumer and A. K. Agogino, “Ensemble clustering with voting active clusters,” *Pattern Recognition Letters*, vol. 29, pp. 1947–1953, October 2008.

[150] C. M. University. (2010) Yippy, Inc. [Online]. Available: http://clusty.com

[151] R. Unnikrishnan, C. Pantofaru, and M. Hebert, “Toward objective evaluation of image segmentation algorithms,” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 929–944, June 2007.

[152] A. v. Wangenheim, R. F. Bertoldi, D. D. Abdala, and M. M. Richter, “Color image segmentation guided by a color gradient network,” *Pattern Recognition Letters*, vol. 28, pp. 1795–1803, 2007.

[153] S. Vega-Pons and J. Ruiz-Schulcloper, “A survey of clustering ensemble algorithms,” *International Journal of Pattern Recognition and Artificial Intelligence*, 2010.

[154] S. Vega-Pons, J. Correa-Morris, and J. Ruiz-Shulcloper, “Weighted partition consensus via kernels,” *Pattern Recognition*, vol. 43(8), pp. 2712–2724, 2010.

[155] K. Wagstaff, “Intelligent clustering with instance-level constraints,” Ph.D. dissertation, Cornell University, 2002.

[156] K. Wagstaff and C. Cardie, "Clustering with instance-level constraints," in *17th International Conference on Artificial Intelligence for Machine Learning (ICML)*, June-July 2000, pp. 1103–1110, clustering.

[157] K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl, "Constrained k-means clustering with background knowledge," *Proceedings of the Eighteenth Conference on Machine Learning*, pp. 577–584, 2001.

[158] K. L. Wagstaff, "When is constrained clustering beneficial, and why," in *Association for the Advancement of Artificial Intelligence*, 2006.

[159] S. Wakana, H. Jiang, L. M. Nagae-Poetscher, P. C. M. van Zijl, and S. Mori, "Fiber tractbased atlas of human white matter anatomy," *Radiology*, vol. 230, no. 1, pp. 77–87, 2004.

[160] R. H. Wasserman, *Tensors and manifolds: With applications to physics.* Oxford University Press, USA, 2004.

[161] P. Wattuya, "Combination of multiple image segmentations," Ph.D. dissertation, Westfälischen Wilhelms-Universität Münster, 2010.

[162] P. Wattuya, K. Rothaus, J.-S. Praßni, and X. Jiang, "A randomwalker based approach to combining multiple segmentations," in *Proceedings of the 19th International Conference on Pattern Recognition*, 2008.

[163] A. Weingessel, E. Dimitriadou, and K. Hornik, "Voting-merging: An ensemble method for clustering," in *In Proceedings of the International Conference on Artificial Neural Networks.* Springer Verlag, 2001, pp. 217–224.

[164] C.-F. Westin and S. E. Maier, "A dual tensor basis solution to the stejskal-tanner equations for DT-MRI," *International Society for Magnetic Resonance in Medicine*, vol. 10, 2002.

[165] C.-F. Westin, S. E. Maier, B. Khidhir, P. Everett, F. A. Jolesz, and R. Kikinis, "Image processing for diffusion tensor magnetic resonance imaging," in *Medical Image Computing and Computer-Assisted Intervetion - MICCAI 1999. Second International Conference.*, 09 1999, pp. 441–452.

[166] R. C. Wilson, I. C. Society, E. R. Hancock, and B. Luo, "Pattern vectors from algebraic graph theory," *IEEE PAMI*, vol. 27, pp. 1112–1124, 2005.

[167] E. Xing, A. Ng, M. Jordan, and S. Russel, "Distance metric learning, with application to clustering with side-information," *NIPS*, 2003.

[168] Q. Xu and M. Desjardins, "Constrained spectral clustering under a local proximity structure assumption," in *In Proceedings of the 18th International Conference of the Florida Artificial Intelligence Research Society.* AAAI Press, 2005.

[169] Q. Xu, M. Desjardins, and K. L. Wagstaff, "K.l.: Active constrained clustering by examining spectral eigenvectors," in *In: Proceedings of the Eighth International Conference on Discovery Science*, 2005, pp. 294–307.

[170] R. Xu and D. W. II, "Survey on clustering algorithms," *IEEE Transactions on Neural Networks*, vol. 16, pp. 645–678, 2005.

[171] K. Y. Yeung, K. Y. Yeung, D. R. Haynor, D. R. Haynor, W. L. Ruzzo, and W. L. Ruzzo, "Validating clustering for gene expression data," *Bioinformatics*, vol. 17, pp. 309–318, 2000.

[172] P. N. Yianilos, "Data structures and algorithms for nearest neighbor search in general metric spaces," in *Proc. of the 4th annual Symp. on Discrete Algorithms*, 1993, pp. 311–321.

[173] H.-S. Yoon, S.-H. Lee, S.-B. Cho, and J. Kim, "A novel framework for discovering robust cluster results," in *Discovery Science*, ser. Lecture Notes in Computer Science, L. Todorovski, N. Lavrac, and K. Jantke, Eds., vol. 4265. Springer Berlin / Heidelberg, 2006, pp. 373–377.

[174] Z. Yu, S. Zhang, H.-S. Wong, and J. Zhang, "Image segmentation based on cluster ensemble," in *Advances in Neural Networks*, ser. LNCS, vol. 4493. Springer Berlin / Heidelberg, 2007, pp. 894–903.

[175] O. Zamir and O. Etzioni, "Grouper: a dynamic clustering interface to web search results," *Computer Networks*, vol. 31, no. 11-16, pp. 1361–1374, May 1999.

[176] S. Zhang and D. H. Laidlaw, "DTI fiber clustering and cross-subject cluster analysis," in *International Society of Magnetic Resonance in Medicine*, 2005.

[177] X. Zhang, L. Jiao, F. Liu, L. Bo, and M. Gong, "Spectral clustering ensemble applied to texture features for sar image segmentation," in *IEEE Transactions on Geoscience and Remote Sensing*, vol. 46, 2008, pp. 2126–2135.

[178] L. Zhengdong and M. A. Carreira-Perpinan, "Constrained spectral clustering through affinity propagation," in *IEEE Conference on Computer Vision and Pattern Recognition.* IEEE, June 2008, pp. 1–8.