A New Graph-Theoretic Approach to Clustering, with Applications to Computer Vision

Massimiliano Pavan

Technical Report UBLCS-2004-04

March 2004

Department of Computer Science
University of Bologna
Mura Anteo Zamboni 7
40127 Bologna (Italy)

The University of Bologna Department of Computer Science Research Technical Reports are available in gzipped PostScript format via anonymous FTP from the area `ftp.cs.unibo.it:/pub/TR/UBLCS` or via WWW at URL `http://www.cs.unibo.it/`. Plain-text abstracts organized by year are available in the directory `ABSTRACTS`. All local authors can be reached via e-mail at the address *lastname*`@cs.unibo.it`.

Recent Titles from the UBLCS Technical Report Series

2002-9 *Towards Adaptive, Resilient and Self-Organizing Peer-to-Peer Systems*, Montresor, A., Meling, H., Babaoglu, O., September 2002.

2002-10 *Towards Self-Organizing, Self-Repairing and Resilient Distributed Systems*, Montresor, A., Babaoglu, O., Meling, H., September 2002 (Revised November 2002).

2002-11 *Messor: Load-Balancing through a Swarm of Autonomous Agents*, Montresor, A., Meling, H., Babaoglu, O., September 2002.

2002-12 *Johanna: Open Collaborative Technologies for Teleorganizations*, Gaspari, M., Picci, L., Petrucci, A., Faglioni, G., December 2002.

2003-1 Security and Performance Analyses in Distributed Systems (Ph.D Thesis), Aldini, A., February 2003.

2003-2 Models and Types for Wide Area Computing. The calculus of Boxed Ambients (Ph.D. Thesis), Crafa, S., February 2003.

2003-3 MathML Formatting (Ph.D. Thesis), Padovani, L., February 2003.

2003-4 Performance Evaluation of Mobile Agents Paradigm for Wireless Networks (Ph.D. Thesis), Al Mobaideen, W., March 2003.

2003-5 Synchronized Hypermedia Documents: a Model and its Applications (Ph.D. Thesis), Gaggi, O., March 2003.

2003-6 Searching and Retrieving in Content-Based Repositories of Formal Mathematical Knowledge (Ph.D. Thesis), Guidi, F., March 2003.

2003-7 Intersection Types, Lambda Abstraction Algebras and Lambda Theories (Ph.D. Thesis), Lusin, S., March 2003.

2003-8 Towards an Ontology-Guided Search Engine, Gaspari, M., Guidi, D., June 2003.

2003-9 An Object Based Algebra for Specifying A Fault Tolerant Software Architecture, Dragoni, N., Gaspari, M., June 2003.

2003-10 A Scalable Architecture for Responsive Auction Services Over the Internet, Amoroso, A., Fanzieri F., June 2003.

2003-11 WSSecSpaces: a Secure Data-Driven Coordination Service for Web Services Applications, Lucchi, R., Zavattaro, G., September 2003.

2003-12 Integrating Agent Communication Languages in Open Services Architectures, Dragoni, N., Gaspari, M., October 2003.

2003-13 Perfect load balancing on anonymous trees, Margara, L., Pistocchi, A., Vassura, M., October 2003.

2003-14 Towards Secure Epidemics: Detection and Removal of Malicious Peers in Epidemic-Style Protocols, Jelasity, M., Montresor, A., Babaoglu, O., November 2003.

2003-15 Gossip-based Unstructured Overlay Networks: An Experimental Evaluation, Jelasity, M., Guerraoui, R., Kermarrec, A-M., van Steen, M., December 2003.

2003-16 Robust Aggregation Protocols for Large-Scale Overlay Networks, Montresor, A., Jelasity, M., Babaoglu, O., December 2003.

2004-1 A Reliable Protocol for Synchronous Rendezvous (Note), Wischik, L., Wischik, D., February 2004.

Dottorato di Ricerca in Informatica

Università di Bologna, Padova, Venezia

A New Graph-Theoretic Approach to Clustering, with Applications to Computer Vision

Massimiliano Pavan

March 15, 2004

Coordinatore:	Tutore:
Prof. Özalp Babaoğlu	Prof. Marcello Pelillo

Abstract

This work applies cluster analysis as a unified approach for a wide range of vision applications, thereby combining the research domain of computer vision and that of machine learning. Cluster analysis is the formal study of algorithms and methods for recovering the inherent structure within a given dataset. Many problems of computer vision have precisely this goal, namely to find which visual entities belong to an inherent structure, e.g. in an image or in a database of images. For example, a meaningful structure in the context of image segmentation is a set of pixels which correspond to the same object in a scene. Clustering algorithms can be used to partition the pixels of an image into meaningful parts, which may correspond to different objects. In this work we focus on the problems of image segmentation and image database organization. The visual entities to consider are pixels and images, respectively.

Our first contribution in this work is a novel partitional (flat) clustering algorithm. The algorithm uses pairwise representation, where the visual objects (pixels, images) are mapped to the nodes of an undirected edge-weighted graph, with edge weights representing pairwise similarity relations. The clustering problem is formulated as a graph partitioning problem, namely a meaningful partitioning of the nodes into clusters is thought.

Our clustering algorithm is centered around the notion of a *dominant set*, a novel combinatorial concept which arises from the study of a continuous formulation of the maximum clique problem. Our proposal seems to be a plausible generalization of the notion of a maximal clique in the context of edge-weighted graphs since, in the unweighted case, dominant sets turn out to be equivalent to (strictly) maximal cliques. Formal properties, intuitive arguments, and empirical findings make dominant sets reasonable candidates for a new

formal definition of a cluster. We also establish a correspondence between dominant sets and the extrema of a quadratic form over the standard simplex. Computationally, this allows us to find clusters using straightforward continuous optimization techniques such as *payoff-monotonic game dynamics*, a class of dynamical systems arising in evolutionary game theory. Such systems are attractive as can be coded in a few lines of any high-level programming language, can easily be implemented in a parallel network of locally interacting units (thereby motivating analog VLSI implementations), and offer the advantage of biological plausibility.

Our second contribution in this work is a hierarchical extension of our partitional clustering algorithm. To this end, we consider a family of (continuous) quadratic programs which contain a parameterized regularization term that controls the global shape of the energy landscape. When the regularization parameter is zero the local solutions are known to be in one-to-one correspondence with dominant sets, but when it is positive an interesting picture emerges. We determine bounds for the regularization parameter that allow us to exclude from the set of local solutions those inducing clusters of size smaller than a prescribed threshold. This suggests a new (divisive) hierarchical approach to clustering, which is based on the idea of properly varying the regularization parameter during the clustering process.

Getting back to the computer vision applications, we demonstrate the potential of the approach for the problems of intensity, color and textured image segmentation as well as image database organization.

Acknowledgements

Firstly, I would like to thank my supervisor, Prof. Marcello Pelillo for his guidance throughout the course of this work. My thanks also go to Prof. Alessandro Verri and Prof. Edwin Hancock for their assessments of the thesis.

Above all I would like to thank my wife for all the support and patience she has given me, especially during the final rush of this writing.

Contents

List of Figures

Chapter 1

Introduction

We present the goal of our work in Section 1.1, and some fundamental problems of computer vision in Section 1.2. Background on clustering algorithms is provided in Sections 1.3 and 1.4. We outline our contribution in Section 1.5, where we also describe the organization of the rest of this thesis.

1.1 The Motivation and the Goal of This Work

A wide range of tasks in computer vision may be viewed as unsupervised partitioning of data. Image segmentation and image database organization are problems at different levels of visual information processing (low and high level vision, respectively). These tasks have different different application objectives, and they handle very different entities of data (pixels or images). Nevertheless, they all reduce to the task of partitioning the visual entities into "coherent" parts.

The practice of classifying objects (or patterns) according to perceived similarities is the basis for much of cognitive science, and beyond [99, 116, 111]. Organizing data into sensible groupings is one of the most fundamental modes of understanding and learning, both for humans and "smart" machines. *Cluster analysis* is the formal study of algorithms and methods for grouping objects. An object is described either by a set of measurement or by relationships between the object and other objects.

A cluster is comprised of a number of *similar* objects grouped together. We document the following definitions of a cluster [53, 22]:

1. "A cluster is a set of entities which are alike, and entities from different clusters are not alike."

2. "A cluster is an aggregation of points in the test space such that the distance between any two points in the cluster is less than the distance between any point in the cluster and any point not in it."

3. "Clusters may be described as connected regions of a multi-dimensional space containing a relatively high density of points, separated from other such regions by a region containing a relatively low density of points."

The last two definitions assume that the objects to be clustered are represented as points in the measurement space. Referring to the first definition, which is more relevant to the current work, the coherence of a cluster should reflect intra-cluster homogeneity and inter-cluster separability.

The goal of this work is to use cluster analysis as a unifying principle for a wide range of vision problems. In our approach, the visual entities are described by relationships with other visual entities. The affinity relationships are a function of the relevant attributes of the entities. Low level attributes might be the spatial location, intensity level, color composition or filter response of a pixel in the image. High level attributes may be as complex as the entire shape of an object in the scene, or the color distribution of all the pixels in an image. A cluster of pixels in the image, sharing similar locations and colors, is expected to account for an object or a part of an object in the scene. A cluster of images in a database is expected to be related with a common topic.

An attractive broad view of vision is that it is an inference problem [27]: we have some measurements, and we wish to determine what caused them. There are crucial features that distinguish vision from many other inference problems: firstly, there is a prohibitively large amount of raw data, and secondly, we don't know which of these data items come from objects, and so help with solving the inference problem, and which do not. This problem can be addressed by working with a compact representation of the interesting image data that emphasizes the properties that make it interesting. Obtaining this representation is known as *segmentation*.

In addition to the problem of compact representation of the raw data there is the problem of efficiently processing the awful amount of data. Algorithms that use low level features can benefit from dividing the raw image data into parts that can be processed separately. Image segmentation is therefore also motivated by the need to design more efficient algorithms.

In the analysis of image databases, a related problem appears. Since we are typically dealing with a large number of images, it is often necessary to divide them into subsets that can be processed separately during image retrieval. Given a high level similarity measure for images and a clustering algorithm, it is possible to organize an image database into categories of interest. A further application of such an organization is to present the content of the database in a concise form, which can be digested by the user [7].

1.2 Segmentation and Image Database Organization

Segmentation and image database organization are fundamental problems of computer vision. An exhaustive survey in these areas are beyond the scope of this section. However, to put our work into context, we focus on some of the more recent or classical methods. Many image segmentation algorithms are reviewed in [78, 15, 27].

The classical techniques include gray level thresholding, region growing and recursive splitting, relaxation by the Markov Random Field approach (originated in the work of [36]), relaxation by neural networks, edge-detection-based algorithms [35], and variational formulations (e. g., [9, 75]).

More recently there has been an increasing interest in segmentation algorithms based on pairwise clustering. This includes the usage of objective functions which are minimized by graph-theoretic methods [120, 112, 71], deterministic annealing [50], agglomerative algorithms [24], multi grid techniques [107], and non local Markov chains.

Orthogonal to these approaches, which try to minimize a certain cost function, are the methods which are built upon local and greedy aggregation [23, 17], upon probability distributions over partitions [32]. or upon spectral decomposition of the similarity matrix [118].

A wide class of image database organization systems is based on image clustering. The process of grouping images can be thought of a natural extension of the organization

principles of lower vision levels [32]. One of the main differences between image database organization systems that are based on image grouping lies in the image representation itself, whether it entails global image information or contains local information as well [7].

The system presented by Chen et al. [16], which is based on global image representation including global color, texture and edge histograms, uses both divisive and agglomerative clustering algorithms. Abdel-Mottaleb et al. [1, 62] presented a technique for image retrieval from large databases based on local color features. A hierarchical model which imposes coarse to fine structure on the image collection is suggested by Bernard et al. [4, 5] and Gdalyahu et al. [32]. A probabilistic and continuous framework for image clustering was recently introduced by Greenspan et al. [43]. They also have recently proposed a new approach based on the information bottleneck principle [38, 39].

A rather different class of image database organization methods is based on supervised grouping, which uses human intervention (see, e.g., [52, 108, 14, 42]).

1.3 Properties of Clustering Algorithms

Clustering algorithms are employed in diverse variety of problems, and consequently show large divergence themselves. The principal properties that distinguish different clustering algorithms are listed in the short survey below. In the next section we concentrate on the class of pairwise algorithms, which are more relevant to the current work.

Data Representation

Clustering algorithms are matched to data type and different types of data lead to different representations. The data items can be mapped to some real normed vector space (called *feature space*), can be described by histograms over a set of features, or can be mapped (possibly implicitly) to the nodes of an edge-weighted graph, with edge weights representing similarity or dissimilarity relations [13].

In the first case, each object in a set of n objects is represented by a set of d attributes and each object is represented by a pattern, or d-dimensional vector [54]. The set itself is viewed as a $n \times d$ *pattern matrix*. This form has the advantage that the data can be easily visualized, which is important since in many applications the data analyst often relies on visual inspection to recognize correlations and deviations from randomness. However,

feature-based approaches have the disadvantage of requiring feature selection, which is often an elusive task. Moreover, they suffer from the inadequacy of the metric utilized in parameter space to appropriately represent (dis)similarities between data items.

While vectorial data characterize an object by a number of attributes, distributional data of an object are described by an empirical probability distribution or histogram over the set of features [90]. Basically, n histograms in $\mathbf{R}^d$ represent the co-occurrence of n objects and d features.

The third form is called "pairwise representation" [49] and although it lacks geometrical notions such as scatter and centroids, it is attractive as no feature selection is required and pairwise relations are also allowed to violate metric properties such as the triangular inequality, a situation that cannot be modeled when data are embedded in a normed space. The same is true with respect to symmetry violation, which can be represented by a directed graph.

Parametric Modeling

The assumption which underlies the modeling approach is that the data to be clustered is generated by some statistical source, e.g. mixture model in which one assumes the data were generated by multiple processes and estimates the parameters of the processes and the number of components in the mixture. The clustering problem is then reduced to a parameter estimation problem, which may addressed by iterative optimization techniques. Classical examples are the k-means, and more generally, the EM algorithms [19]. Despite the elegance of this approach, the estimation process leads to a notoriously difficult optimization. For example, the frequently used EM algorithm often converges to a local maximum that depends on the initial conditions.

Exclusive versus nonexclusive

An exclusive (hard) classification is a partition of the set of objects. Each object belongs to exactly one cluster. Nonexclusive classification can assign an object to several clusters. Fuzzy clustering is a type of nonexclusive classification in which a pattern is assigned a degree of belongingness to each cluster in a partition [53].

Hierarchical versus Partitional

A hierarchical classification is a nested sequence of partitions whereas a partitional classification is a single partition. An hierarchy of solutions can be thought of a multi scale interpretation of the data.

An agglomerative, hierarchical clustering places each object in its own cluster and gradually merges these atomic clusters into larger and larger clusters until all objects are in a single cluster. Divisive, hierarchical clustering reverses the process by starting with all objects in one cluster and subdividing into smaller pieces.

Deterministic versus Stochastic

This issue is most relevant to partitional approaches designed to optimize a cost function. This optimization can be accomplished using traditional techniques or through a random search of the state space consisting of all possible labelings. Deterministic search techniques may guarantee an optimal partition by performing exhaustive enumeration. On the other hand, the stochastic search techniques generate a near-optimal partition reasonably quickly, and guarantee convergence to optimal partition asymptotically.

Compactness versus Separation versus Connectedness

Different clustering algorithms seek clusters having different structures. Various algorithms favor compact structures, e.g., the k-means, the complete-link and maximal-clique-based algorithms [53, 3, 96], the pairwise and histogram data clustering formulations by Buhmann et al. [49, 94]. Such methods stress high mutual similarity of cluster elements. A different class of algorithms stresses connectedness of cluster elements via mediating elements rather than favoring high mutual similarity. Examples are the single-link, the minimum spanning tree and connected-component-based algorithms [53], and the recently introduced path-based clustering [24]. These algorithms explicitly favor chained and elongated structures. Finally, the family of cut-based algorithms [120, 112] inherently seek well-separated clusters (see below).

1.4 Review on Pairwise Clustering

Pairwise data clustering methods map (possibly implicitly) the data items to be clustered to the nodes of a weighted graph, with edge weights representing similarity or dissimilarity relations. This section briefly reviews clustering methods which are formulated as graph partitioning.

Embedding

Metric dissimilarity relations d_{ij} between datapoints may be transformed to vectorial representation. Consequently, instead of solving the graph partitioning problem, one faces with the embedding problem followed by the application of vectorial clustering (using an algorithm which may use such geometrical notions as centroids and scatter). The task of embedding given dissimilarity data d_{ij} in a d-dimensional Euclidian space, which is also a prerequisite for visual inspection, is known as multidimensional scaling [20, 57] and it is usually formulated as an optimization problem [63, 61]. The embedding might preserves the ranking of the pairwise distances, not necessarily their ratios.

An alternative method uses the mapping that assigns to each datapoint i the vector $[d_{i_1}, d_{i_2}, \ldots, d_{i_n}]$. Such a mapping embeds n points in $\mathbf{R}^n$ with the max norm l_∞. However, n is usually very large and dimensionality reduction is often required, where possible methods are principal component analysis [29], random projection [56], principal curves [45] and others, possibly introducing distortion into the low dimensional representation. Dimensionality reduction is closely related to the problem of feature selection, namely it involves the assumption that a small number of features can be found, which describe the datapoints with sufficient accuracy.

Heuristics

Probably the best known heuristic for pairwise clustering is agglomeration, which starts from the trivial partition of n points into n clusters of size one, and continue by subsequently merging the two most similar clusters. Different similarity measures between clusters distinguish between different agglomerative algorithms. In particular, the single-link algorithm defines the similarity between clusters as the maximal similarity between two of their members, and the complete linkage algorithm uses the minimal such value.

A very general class of agglomerative algorithms is formalized in [64, 53]. Agglomerative heuristics lack the ability to separate between the problem definition and algorithmic solution. Nevertheless, greedy agglomeration can also be connected with more global properties. For example, the single- and complete-link heuristics are closely related to the minimum spanning tree algorithm and maximal-clique-based algorithms respectively [53]. Furthermore, an elegant connection between agglomeration and cut-based algorithms has recently been established by Gdalyahu et al. [32] (see also [10]). Their (stochastic) *typical cut* algorithm induces a probability distribution over partitions and compute an "average" cut of the data. The typical cut algorithm uses Karger's contraction algorithm [58], which is indeed a randomized version of the single-link agglomerative heuristic. See [23, 76] for additional examples of greedy agglomeration with global properties.

Another well motivated heuristic is spectral decomposition (see below).

Objective Function

The clustering problem can be formulated as a discrete optimization problem, in which case it involves two distinct steps: (i) determine some suitable cost function over the partition (or hierarchy) space; (ii) compute a partition (or hierarchy) which minimizes the cost. However, defining a meaningful cost is task dependent, and the search for its global extremum is in general computationally intractable.

The minimal cut clustering algorithm defines the optimal partition as the one with minimal capacity [55, 120]. For bipartitions, which split the data into two parts, this optimization problem is extensively studied and is solved in polynomial time. However, clustering applications frequently involve very large graphs, and the exact solution becomes impractical. A major disadvantage of the capacity (cut) cost function, namely that it is biased towards partitions which cuts a small number of edges, and may even separate a single node from the graph, led to the proposal of another cost function which is called "normalized cut" ([112], see below).

Various cost functions for clustering have been proposed in [20]. Herault and Horaud [46] formalized foreground/background discrimination as a combinatorial optimization problem. The model assumes that all foreground objects are similar to each other. The use of deterministic annealing in pairwise clustering was explored in Hofmann and Buhmann [49]. They introduced a cost function which stresses cluster coherency and is

based on the idea of summing (and normalizing) all intra-cluster dissimilarities. Recently, a cost function with an explicit bias for chained structures has been recently proposed in [24], where an agglomerative algorithm is used for optimization. We refer the reader to [95] for a classification theory of pairwise optimization-based clustering algorithms.

Cost functions for the pairwise data clustering problem have also been derived in probabilistic settings. Folded Markov chains are used in [68] to derive various cost functions and are thoroughly analyzed in [8, 41]. In [41] it is also shown that the cost function for pairwise clustering introduced in [49] has an interesting connection with Bayes' theorem. Related work include also [2].

Cost functions can be also defined over partition trees (i.e., hierarchies) instead of over single partitions [73, 100, 95, 122].

Spectral methods

Spectral methods identify good partitions via the eigenvectors of the similarity matrix, or other matrices derived from it. These approaches are attractive in that they are based on simple algorithms whose stability is well understood. They are closely related and all use dominant eigenvectors of matrices to perform clustering. However, these approaches use different matrices, focus on different eigenvectors and use a different method of going from the continuous eigenvectors to the discrete clustering.

Perona and Freeman [91] suggested a clustering algorithm (known as the "factorization method") based on treating as an indicator function the first eigenvector ($\mathbf{v}_1$) of the similarity matrix A. A threshold T is chosen, and each node i is assigned to one part if $v_{1_i} > T$ and to the other part otherwise. Perona and Freeman motivated the approach by showing that for block diagonal affinity matrices, the first eigenvector has nonzero components corresponding to points in the dominant cluster and zeros in components corresponding to points outside the dominant cluster. All this is closely related to the approach suggested by Sarkar and Boyer [104] in the context of change detection. They argued that a maximally cohesive cluster can be found by maximizing the quadratic form $\mathbf{x}^{\mathrm{T}} A\mathbf{x}$ over the sphere, and showed that the maximizers are indeed the eigenvectors of A. Recently, Robles-Kelly and Hancock [97] have embedded Sarkar and Boyer's approach in a probabilistic framework deriving an EM-like algorithm for clustering. Related work include also [60], where possibly asymmetric similarity matrices are considered, [98], where

a fast approximation of the first eigenvector is proposed and [92, 34] from the bibliometrics domain.

Shi and Malik [112] used a quite different eigenvector for solving clustering problems. Rather than examining the first eigenvector of A they look at generalized eigenvectors. Let $D = (d_{ij})$ be the degree matrix of A:

$$d_{ii} = \sum_j a_{ij} \tag{1.1}$$

Define the generalized eigenvector $\mathbf{y}$ as a solution to:

$$(D - A)\mathbf{y} = \lambda D\mathbf{y} \tag{1.2}$$

and define the second generalized eigenvector, denoted $\mathbf{y}_2$, as the $\mathbf{y}$ corresponding to the second *smallest* λ. Shi and Malik suggested thresholding this second generalized eigenvector of A in order to partition the nodes into two parts. If $y_{2_i} > T$ then node i is in one part, otherwise it is in the other. Shi and Malik motivated the approach by showing that the second generalized eigenvector is an approximate solution to a continuous version of a *discrete* problem in which the goal is to minimize:

$$\frac{\mathbf{y}^{\mathrm{T}}(D - A)\mathbf{y}}{\mathbf{y}^{\mathrm{T}}D\mathbf{y}} \tag{1.3}$$

subject to the constraint that $\mathbf{y}_i \in \{1, -b\}$ and $\mathbf{y}^{\mathrm{T}}D\mathbf{e} = 0$, where $\mathbf{e}$ is a vector of appropriate length consisting of unit entries and b a positive real constant.

The significance of the discrete problem is that its solution can be shown to provide the partition that minimizes the *normalized cut* cost function:

$$\mathrm{Ncut}(S, T) = \frac{\mathrm{cut}(S, T)}{\mathrm{asso}(S, V)} + \frac{\mathrm{cut}(S, T)}{\mathrm{asso}(T, V)} \tag{1.4}$$

where V is the set of nodes being split into two disjoint sets S, T, $\mathrm{cut}(S, T) = \sum_{i \in S, j \in T} a_{ij}$ and $\mathrm{asso}(S, V) = \sum_j \sum_{i \in S} a_{ij}$. Since S and T are disjoint, each numerator is the cut capacity (sum over edges which cross the cut). In the denominator the terms $\mathrm{asso}(S, V)$ and $\mathrm{asso}(T, V)$ are called the associations of S and T, respectively. They act as penalties for unbalanced cuts.

Thus the solution to the discrete problem finds a bipartitions that minimizes the affinity between groups normalized by the affinity within each group. As Shi and Malik noted,

there is no guarantee that the real solution will bear any relationship with the discrete one. Indeed, they showed that the discrete optimization of equation (1.3) is NP-complete.

Note that while Perona and Freeman use the *largest* eigenvector, Shi and Malik use the second *smallest* generalized eigenvector. Thus the two approaches appear quite different. There is, however, a close connection. See [118] for the relation between the second generalized eigenvector $\mathbf{y}_2$ and the principal eigenvector of $D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$. Interestingly, the factorization method, which seeks tight clusters, furnishes an approximate solution to the problem of finding a partition (S, T) which maximizes:

$$\frac{\text{asso}(S, S)}{|S|} + \frac{\text{asso}(T, T)}{|T|}$$

Both the normalized cut and the factorization algorithm can be motivated by a continuous approximation to a discrete optimization problem. However, no guarantee is provided on the quality of the approximation. Generally speaking, spectral methods are not associated with any exact global cost function and can be thought of useful heuristics.

Many other spectral approaches have been proposed in the literature (e.g., [105, 33, 18, 77]) and we refer the reader to [118, 77] for a review.

1.5 Our Contribution, and the Thesis Organization

In this work we present a novel clustering algorithm, and apply it to problems of image segmentation and image database organization. We first introduce our method for partitional clustering and then extend it to hierarchical clustering.

Our clustering algorithm, which uses pairwise representation, is centered around the notion of a *dominant set*, a novel combinatorial concept which arises from the study of a continuous formulation of the maximum clique problem. Our proposal seems to be a plausible generalization of the notion of a maximal clique in the context of edge-weighted graphs since, in the unweighted case, dominant sets turn out to be equivalent to (strictly) maximal cliques. Formal properties, intuitive arguments, and empirical findings make dominant sets reasonable candidates for a new formal definition of a cluster in the context of edge-weighted graphs. We also establish a correspondence between dominant sets and the extrema of a quadratic form over the standard simplex. Computationally, this allows us to find clusters using straightforward continuous optimization techniques such as *payoff-monotonic game dynamics*, a class of dynamical systems arising in evolutionary game.

Such systems are attractive as can be coded in a few lines of any high-level programming language, can easily be implemented in a parallel network of locally interacting units (thereby motivating analog VLSI implementations), and offer the advantage of biological plausibility.

Our second contribution in this work is an hierarchical extension of our partitional clustering algorithm. To this end, we consider a family of (continuous) quadratic programs which contain a parameterized regularization term that controls the global shape of the energy landscape. When the regularization parameter is zero the local solutions are known to be in one-to-one correspondence with dominant sets, but when it is positive an interesting picture emerges. We determine bounds for the regularization parameter that allow us to exclude from the set of local solutions those inducing clusters of size smaller than a prescribed threshold. This suggests a new (divisive) hierarchical approach to clustering, which is based on the idea of properly varying the regularization parameter during the clustering process.

Thesis Organization

The rest of this work is organized as follows. Chapters 2 to 4 describe our partitional clustering algorithm. The novel notion of a cluster is described in Chapter 2, the continuous formulation of the clustering problem is described in Chapter 3, and the optimization technique we use to solve it is described in Chapter 4. Some results are summarized in Chapter 5, where we apply our partititional clustering algorithm to image segmentation and shape categorization. The hierarchical extension is described in Chapter 6, and results on image database organization are described in Chapter 7. A concluding discussion follows in Chapter 8.

Chapter 2

Clustering by Dominant Sets

Many graph-theoretic clustering algorithms basically consist of searching for certain combinatorial structures in the similarity graph, such as a minimum spanning tree [121] or a minimum cut [32, 112, 120] and, among these methods, a classic approach to clustering (the "complete-link" algorithm [53]) reduces to a search for a complete subgraph, namely a *clique*.[1] Indeed, some authors [3, 96] argue that the maximal clique is the strictest definition of a cluster. Unfortunately, while the minimum spanning tree and the minimum cut (and variations thereof) are notions that are explicitly defined on edge-weighted graphs, the concept of a maximal clique is defined on unweighted graphs, and it is not clear how to generalize it to the edge-weighted case. As a consequence, maximal-clique-based clustering algorithms typically work on unweighted graphs derived from the similarity graph by means of some threshold operation [53, 3, 40]. Although such threshold operations can be used to generate a hierarchy of clusters displayed to a user in the form of a *dendogram* [53], in many computer vision applications this approach is infeasible due to the large number of data to be clustered. It is therefore of considerable interest to extend the notion of a maximal clique to edge-weighted graphs, thereby allowing the development of a new non-hierarchical (partitional) clustering approach.

To this end, we propose a new approach for pairwise clustering based on a novel combinatorial concept (that of a *dominant set*) which arises from the study of a continuous formulation of the maximum clique problem, originally due to Motzkin and Straus [74]. Our proposal seems to be a plausible generalization of the notion of a maximal clique in

[1] Recall that a subset of vertices of a graph is said to be a *clique* if all its nodes are mutually adjacent; a *maximal* clique is one which is not contained in any larger clique, whereas a *maximum* clique is one having largest cardinality.

the context of edge-weighted graphs since, in the unweighted case, dominant sets turn out to be equivalent to (strictly) maximal cliques.

This chapter presents our novel definition of a cluster, which is used in Chapters 5 and 7 for a few vision applications. The general approach and the principles of our method are described first. We introduce our novel definition and the high-level algorithm in Section 2.1. To clarify how the algorithm works and to demonstrate some of its properties, an illustrative example is worked out in details in Section 2.2. In Section 2.3 we investigate the robustness of our method and present some additional examples. Finally, in Sections 2.4 and 2.5 we investigate the relationships between our novel definition of a cluster and that of a (strictly) maximal clique.

2.1 Graph-theoretic Definition of a Cluster

We represent the data to be clustered as an undirected edge-weighted graph with no self-loops $G = (V, E, w)$, where $V = \{1, \ldots, n\}$ is the vertex set, $E \subseteq V \times V$ is the edge set, and $w : E \to \mathbf{R}^*_+$ is the (positive) weight function. Vertices in G correspond to data points, edges represent neighborhood relationships, and edge-weights reflect similarity between pairs of linked vertices. As customary, we represent the graph G with the corresponding *weighted adjacency* (or *similarity*) matrix, which is the $n \times n$ symmetric matrix $A = (a_{ij})$ defined as:

$$a_{ij} = \begin{cases} w(i,j) \ , & \text{if } (i,j) \in E \\ 0 \ , & \text{otherwise.} \end{cases}$$

We do not assume that the similarity weights obey the triangular inequality, and self similarities $w(i,i)$ are not defined.

A common informal definition states that "a cluster is a set of entities which are *alike*, and entities from different clusters are not alike" (see Section 1.1). Hence, a cluster should satisfy two fundamental conditions: (a) it should have high internal homogeneity; (b) there should be high inhomogeneity between the entities in the cluster and those outside. When the entities are represented as an edge-weighted graph, these two conditions amount to saying that the weights on the edges within a cluster should be large, and those on the edges connecting the cluster nodes to the external ones should be small.

To give our formal definition of a cluster, we start with the intuitive idea that the assignment of the edge-weights induces, in some way to be described, an assignment of

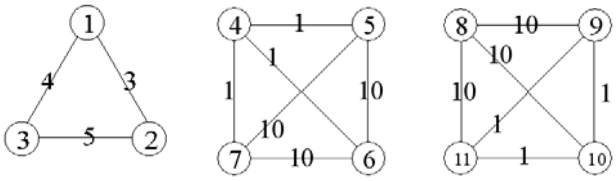


Figure 2.1: An example edge-weighted graph.

weights on the vertices. This perspective gives us a chance to analyze the assignment of the edge-weights in a simpler and fruitful way. To grasp the intuition behind this idea, consider the graph in Figure 2.2 and the subgraph induced by the set $S = \{1, 2, 3\}$. Observe that the edges incident to vertex 1 are the lightest ones (within the subgraph), the heaviest ones are incident to vertex 3 and those incident to 2 are the lightest as well as the heaviest ones. This induces a sort of natural ranking among the vertices of S, which is captured by the notions introduced below.

Let $S \subseteq V$ be a non-empty subset of vertices and $i \in V$. The *(average) weighted degree* of i w.r.t. S is defined as: [2]

$$\mathrm{a}wdeg_S(i) = \frac{1}{|S|} \sum_{j \in S} a_{ij} \ . \tag{2.1}$$

Observe that $\mathrm{a}wdeg_{\{i\}}(i) = 0$ for any $i \in V$. Moreover, if $j \notin S$ we define (see Figure 2.2):

$$\phi_S(i, j) = a_{ij} - \mathrm{a}wdeg_S(i) \ . \tag{2.2}$$

Note that $\phi_{\{i\}}(i, j) = a_{ij}$, for all $i, j \in V$ with $i \neq j$. Intuitively, $\phi_S(i, j)$ measures the similarity between nodes j and i, with respect to the average similarity between node i and its neighbors in S. Note that $\phi_S(i, j)$ can be either positive or negative. Referring to the graph in Figure 2.1 we have:

$$\phi_{\{1,2\}}(1, 3) = a_{13} - \mathrm{a}wdeg_{\{1,2\}}(1) = 4 - \frac{0+3}{2} = \frac{5}{2} \ .$$

We are now in a position to formalize the notion of "induction" of node-weights, which is captured by the following recursive definition.

[2]Note that, although we consider graphs with no self-loops, in computing the average weighted degree of $i \in S$ we divide by $|S|$ instead of $|S|-1$, and this can lead to somehow counterintuitive results especially when S is a small set. For example, referring to the graph in Fig. 2.1 $\mathrm{a}wdeg_{\{1,2\}}(1) = \frac{0+3}{2} < 3$.

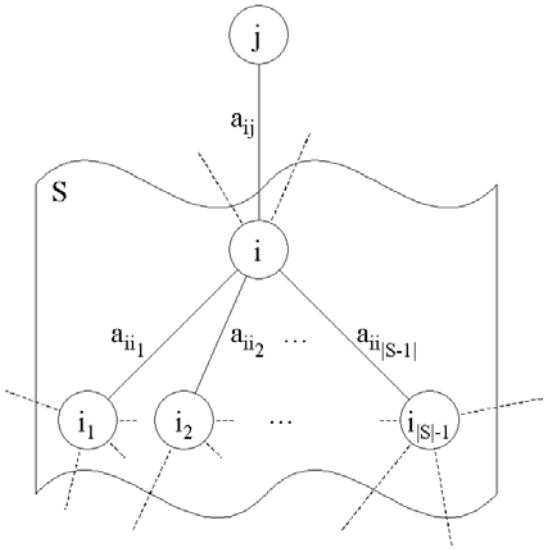


Figure 2.2: Calculation of $\phi_S(i,j)$. Only the nodes and the edges plotted in solid line are involved in the computation of $\phi_S(i,j)$ (here, $S = \{i_1, i_2, \ldots, i_{|S|-1}, i\}$).

Definition 2.1 *Let $S \subseteq V$ be a non-empty subset of vertices and $i \in S$. The* weight *of i w.r.t. S is*

$$\mathrm{w}_S(i) = \begin{cases} 1, & \text{if } |S| = 1 \\ \sum_{j \in S \setminus \{i\}} \phi_{S\setminus\{i\}}(j,i)\, \mathrm{w}_{S\setminus\{i\}}(j), & \text{otherwise.} \end{cases} \tag{2.3}$$

Moreover, the total weight *of S is defined to be:*

$$\mathrm{W}(S) = \sum_{i \in S} \mathrm{w}_S(i) \ . \tag{2.4}$$

Note that $\mathrm{w}_{\{i,j\}}(i) = \mathrm{w}_{\{i,j\}}(j) = a_{ij}$, for all $i, j \in V$ $(i \neq j)$. Also, observe that $\mathrm{w}_S(i)$ is calculated simply as a function of the weights on the edges of the subgraph induced by S. Moreover, the weights so defined respect the intuitive ranking illustrated above. For example, referring again to the graph in Figure 2.1, it turns out that $\mathrm{w}_{\{1,2,3\}}(1) < \mathrm{w}_{\{1,2,3\}}(2) < \mathrm{w}_{\{1,2,3\}}(3)$.

Intuitively, $\mathrm{w}_S(i)$ gives us a measure of the overall similarity between vertex i and the vertices of $S \setminus \{i\}$ with respect to the overall similarity among the vertices in $S \setminus \{i\}$. For example, in the graph of Figure 2.1 it turns out that $\mathrm{w}_{\{4,5,6,7\}}(4) < 0$ and $\mathrm{w}_{\{8,9,10,11\}}(8) > 0$ and this can be intuitively grasped by looking at the amount of edge-weight associated

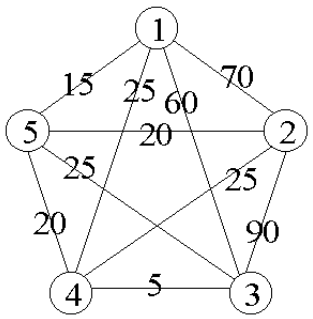


Figure 2.3: The subset of vertices $\{1, 2, 3\}$ is dominant.

to vertices 4 and 8: that associated to vertex 4 is significantly smaller than that of subset $\{5, 6, 7\}$; conversely, that associated to vertex 8 is significantly greater than that of subset $\{9, 10, 11\}$.

The following definition represents our formalization of the concept of a cluster in an edge-weighted graph.

Definition 2.2 *A non-empty subset of vertices $S \subseteq V$ such that* $\mathrm{W}(T) > 0$ *for any non-empty $T \subseteq S$, is said to be* dominant *if:*

1. $\mathrm{w}_S(i) > 0$, *for all* $i \in S$

2. $\mathrm{w}_{S\cup\{i\}}(i) < 0$, *for all* $i \notin S$.

The two conditions of the above definition correspond to the two main properties of a cluster: the first regards internal homogeneity, whereas the second regards external inhomogeneity. The condition $\mathrm{W}(T) > 0$ for any non-empty $T \subseteq S$ is a technicality explained in some detail in Chapter 3.

To illustrate, in the graph of Figure 2.3 the subset of vertices $\{1, 2, 3\}$ is dominant, and this may be intuitively explained by observing that the edge weights "internal" to that set (60, 70 and 90) are larger than those between internal and external vertices (which are between 5 and 25). As the example suggests, the main property of a dominant set is that the overall similarity among internal nodes is higher than that between external and internal nodes, and this fact is the motivation of considering a dominant set as a cluster of nodes.

procedure PARTITIONAL_CLUSTERING:
input: weighted graph $G = (V, E, w)$ with n nodes.
output: a partition $\mathcal{P}$.
$\mathcal{P} = \emptyset$
while $STOPPING\_CRITERION(G)$:
  $S \leftarrow$ DOMINANT_SET(G)
  $\mathcal{P} \leftarrow \mathcal{P} \cup \{S\}$
  $V \leftarrow V \setminus S$
end-loop
return $\mathcal{P}$

Figure 2.4: Pseudo-code for our partitional clustering algorithm.

Basically, our clustering algorithm consists of iteratively finding a dominant set in the graph (i.e., a cluster), and then removing it from the graph, until all vertices have been clustered. The algorithm is described in pseudo-code in Figure 2.1. In this pseudo-code the procedure DOMINANT_SET finds a dominant set in the current graph G. The procedure DOMINANT_SET is the cornerstone of our method and will be described in Chapter 4. The function STOPPING_CRITERION simply check whether the current graph G is empty or not.

The assignment of node weights naturally provides us with a measure of the overall similarity of a dominant set. Given a dominant set $S \subseteq V$, we measure its overall cohesiveness with:

$$\frac{1}{2}\frac{\sum_{i \in S} \mathrm{a}wdeg_S(i)\, \mathrm{w}_S(i)}{\mathrm{W}(S)} \tag{2.5}$$

We shall formally motivate this measure of cohesiveness in Chapter 3.

2.2 Illustrative Example

To illustrate some features of our method, we proceed with an illustrative synthetic example. We use a point set example in two dimensions which can be easily visualized, see Figure 2.5a.

The data consists of $n = 39$ points in $\mathbf{R}^2$ arranged in four globular regions. The vectorial nature of data in this example, namely points in $\mathbf{R}^2$, is used for visualization but

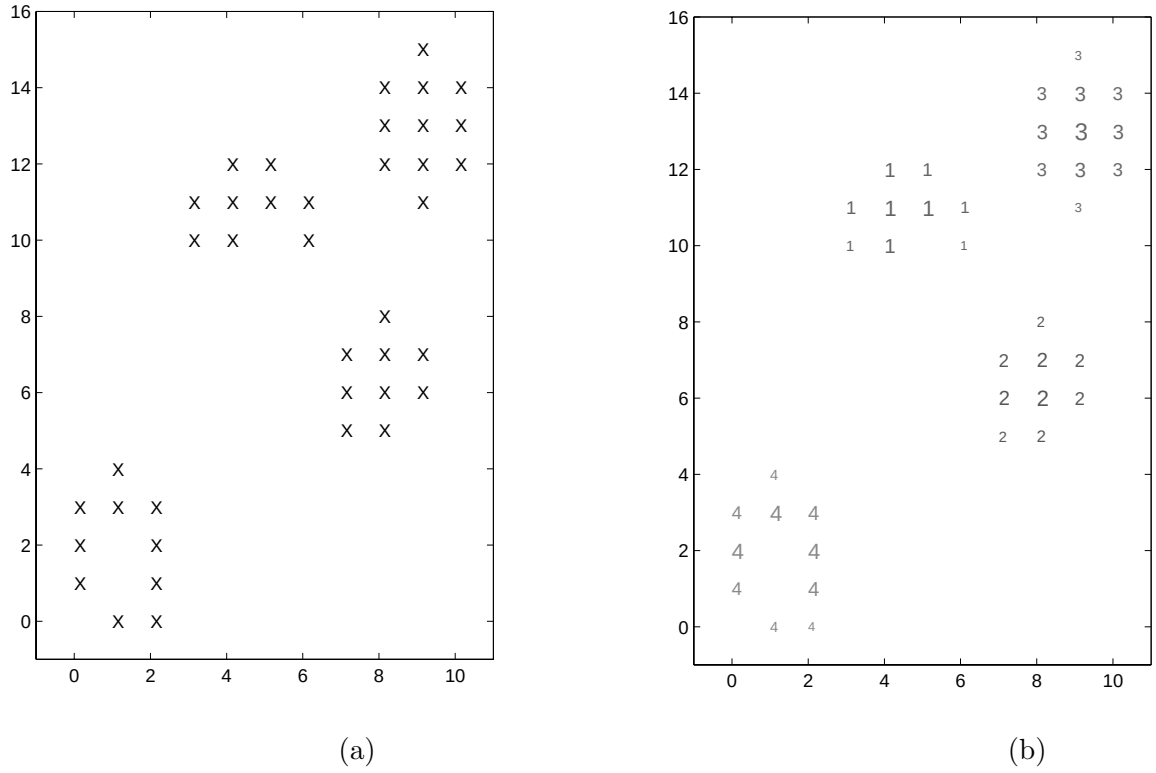


Figure 2.5: Clustering of points in the Euclidean plane. (a) The 39 data points; the coordinates of the points are not available to the clustering algorithm, which uses only the matrix of pairwise distances. (b) Clustering result.

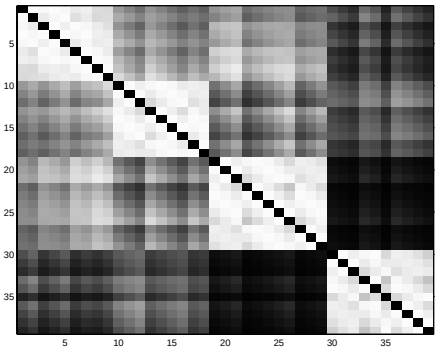


Figure 2.6: The similarity matrix a_{ij} computed for 39 points.

is hidden from the clustering algorithm. The information which is made available to the clustering algorithm includes only the matrix of pairwise Euclidean distances d_{ij} between points.

During preprocessing, the Euclidean distances d_{ij} are transformed to similarity weights, which decrease with increasing distances. Psychophysical studies found that the similarity, or the confusion frequency, between stimuli decay exponentially with some power of the perceptual measure distance [109, 110, 21].

As customary, we use the following functional form:

$$w(i,j) = \exp(-d_{ij}^2/\sigma^2) \ , \tag{2.6}$$

where σ is a positive real number which affects the decreasing rate of w (here we used $\sigma = 7.85$).

We then construct a complete graph $G(V, E, w)$ with n nodes, where node i represents the i-th data point and the weight of edge i, j is set to $w(i, j)$. The resulting similarity matrix is shown in Fig. 2.6. Here, and in the sequel, the gray-levels in the matrix representation are proportional to the similarities: the darker the entries, the weaker the similarities between the corresponding elements. Moreover, data items which belong to the same cluster are ordered next to each other, hence the matrix shows a block diagonal structure.

The constant σ reflects some reasonable local scale. If $d_{ij} \gg \sigma$ then $w(i, j)$ is very small and the nodes i and j are unlikely to belong to the same dominant set. The number

$\sigma = 7.85$ is the only parameter involved in the preprocessing stage. Note that the weights are in the range $[0, 1]$.

Having constructed the similarity graph G, we are ready to apply our clustering algorithm. Figure 2.5b shows a partition of the original data points into 4 clusters. Each cluster is labeled with a number indicating the order of extraction from the graph by the procedure DOMINANT_SET (see pseudocode in Figure 2.1). Here, and in the sequel, the font size in the figures are proportional to the participation: the smaller the size, the weaker the participation of the corresponding point in the cluster.

Figure 2.7a shows again cluster 1 from Figure 2.2 (here, we call it S): the 9 elements of the cluster are ordered in accordance to their participations, measured by $\mathrm{w}_S(i)$ for $i = 1 \ldots 9$. Figure 2.7b shows the ratios $\frac{\mathrm{w}_S(i)}{\mathrm{W}(S)}$ for $i = 1 \ldots 9$. Note that, although the coordinates of the points are not available to the clustering algorithm, the two central elements (1 and 2) have the highest participation values. Furthermore, notice that the farther the point from the barycenter of the cluster, the weaker the participation of the corresponding node in the cluster.

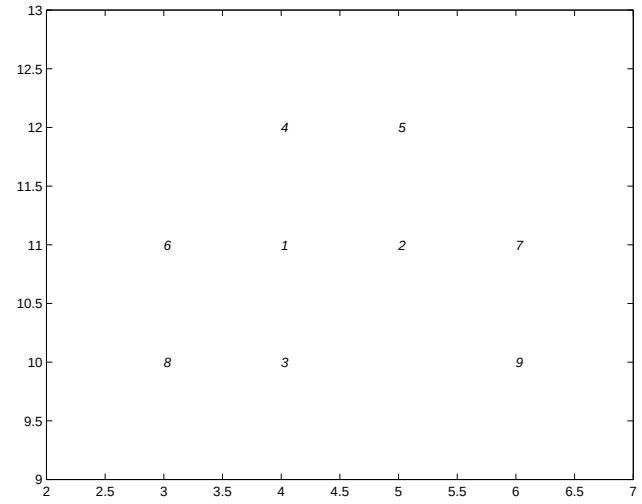
Figure 2.8 shows the cohesiveness values (calculated according to (2.5)) of the four clusters in Figure 2.5b. Not that the most compact cluster (cluster 3) has the highest value of cohesiveness, while the least compact one (cluster 4) has the lowest value.

2.3 Robustness and Additional Examples

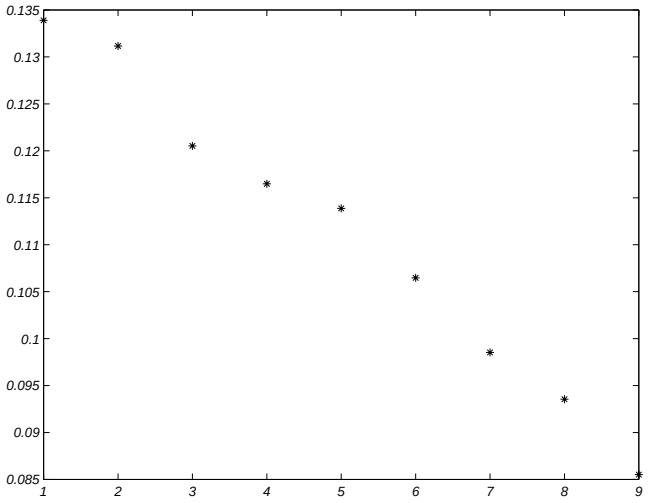
Robustness with respect to external parameters of the algorithm and its sensitivity to their values is discussed here.

It is only the preprocessing stage, which constructs the weighted graph G, that depends on external parameters. These parameters are related to the definition of similarity (which is task dependent), and to transformation from perceptual similarity to edge weight.

Figure 2.3 extends the illustrative example of Section 2.2. We repeat the same experiment using the same weighting function but with different local scale values, setting $\sigma = 5.7, 5.8, 9.9, 10$. The larger σ is, the slower is the decay rate of the weights $w(i, j)$. We conclude that the parameter σ in this example can be varied by almost 30% before the local structure is lost.



(a)



(b)

Figure 2.7: Node participation. (a) The 9 data points of cluster 1 from Figure 2.5b. (b) The participation of the nodes in the cluster.

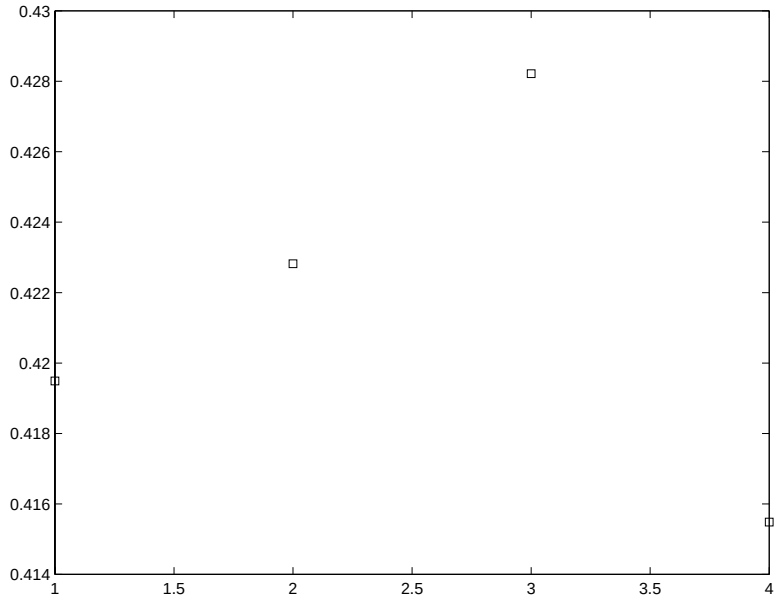
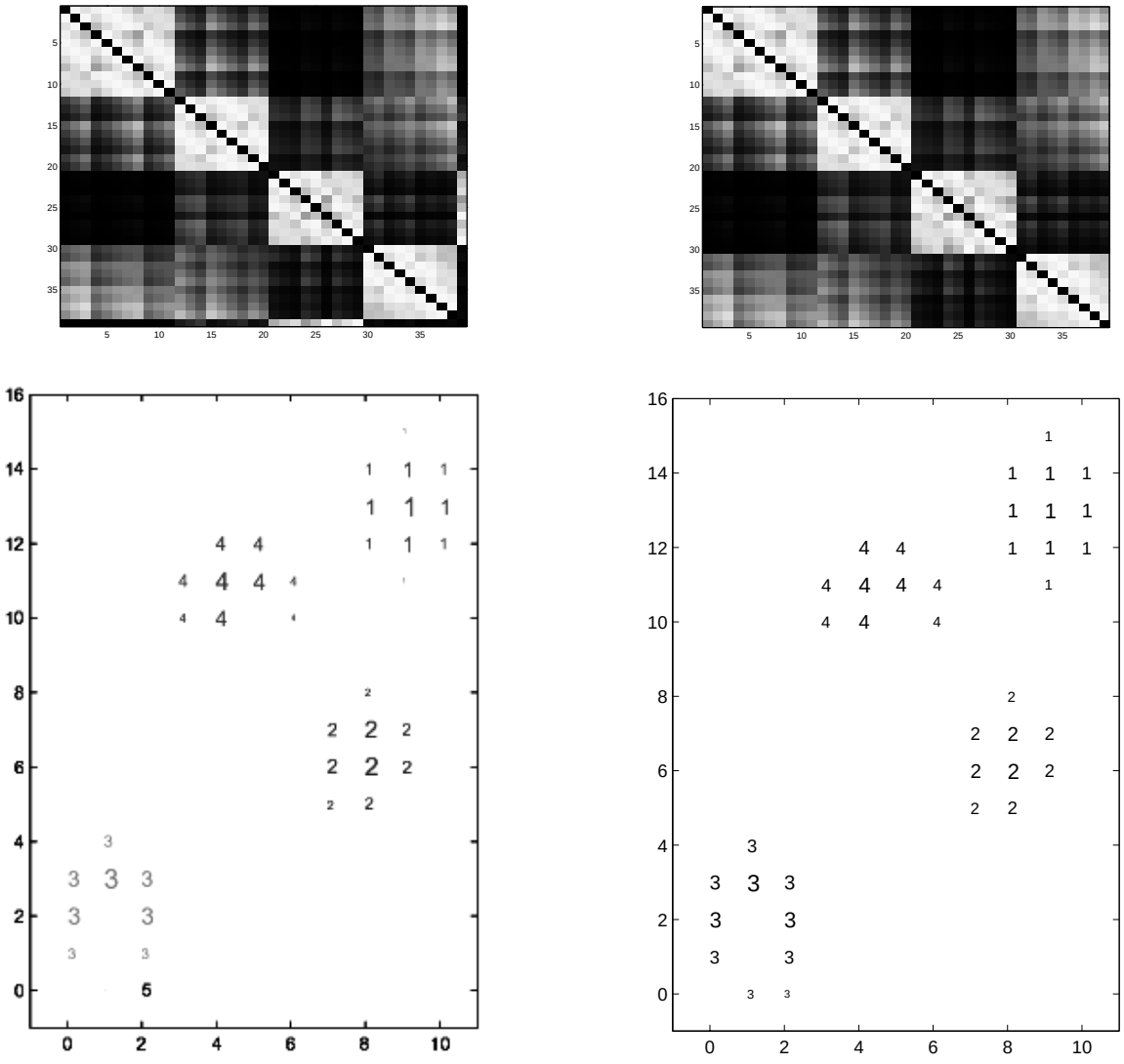
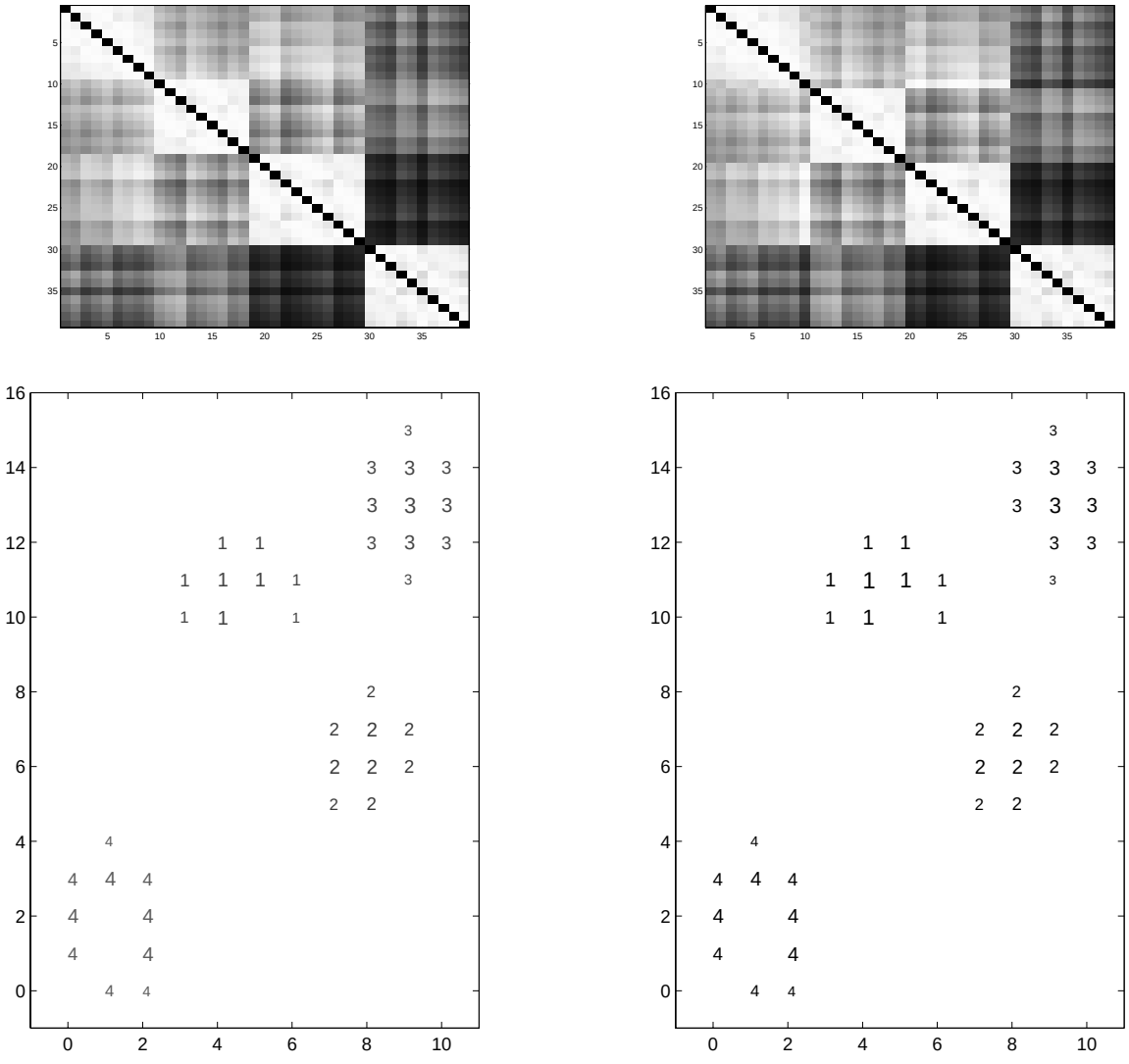


Figure 2.8: Cohesiveness values of the four clusters in Figure 2.5b. The cohesiveness values are calculated according to (2.5).



$\sigma = 5.7$ $\sigma = 5.8$

Figure 2.9: Robustness with respect to the local scale parameter. The local scale (σ) determines the decay rate of the weights $w(i,j)$. We repeat the experiment with $\sigma = 5.7, 5.8, 9.9, 10$.



$\sigma = 9.9$ $\sigma = 10$

Figure 2.10: Robustness with respect to the local scale parameter. The local scale (σ) determines the decay rate of the weights $w(i,j)$. We repeat the experiment with $\sigma = 5.7, 5.8, 9.9, 10$.

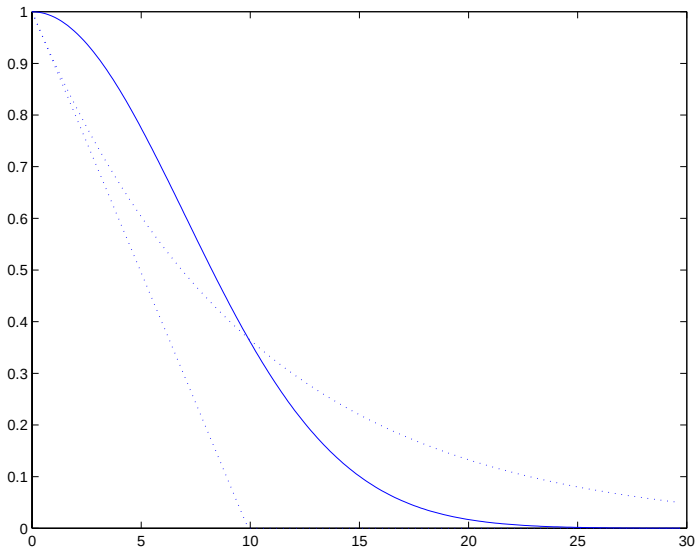


Figure 2.11: The weighting function $w(i,j) = \exp(-d_{ij}^2/9.9^2)$ shown in solid line. The dotted lines show the two alternative weighting functions used in Figures 2.12 and 2.13.

We now investigate the sensitivity to the transformation from perceptual similarity to edge weight. Different clusters often have very different within-group similarity or coherence. It is very difficult to predetermine the right weighting function on each cluster. Therefore, it is important to design a grouping algorithm that is tolerant to a wide range of weighting functions. We use three decreasing weighting functions, with different rate of fall-off, on the point set example in Figure 2.5a. The three weighting functions are plotted in Figures 2.11, 2.12b and 2.13b. The first two functions, $w(i,j) = \exp(-d_{ij}^2/\sigma^2)$ and $w(i,j) = \exp(-d_{ij}/\sigma)$ have a moderate decreasing rate (here $\sigma = 9.9$ was used). The third function, $w(i,j) = \max(1 - d_{ij}/\sigma, 0)$, has the fastest decreasing rate among the three. In all cases, the algorithm was able to correctly partition the data points.

Before concluding this section, we report some additional examples of point set clustering. Referring to the point set in Figure 2.14, the algorithm makes a good job in finding a meaningful clustering. However, a problem arises with the cluster on the top right, whose elongated shape is difficult to be recovered by the algorithm: erroneously, an extra spurious cluster was found. Indeed, our algorithm has a bias for finding tight, blob-like clusters.

As an additional example, consider the point set in Figure 2.15. Here the algorithm

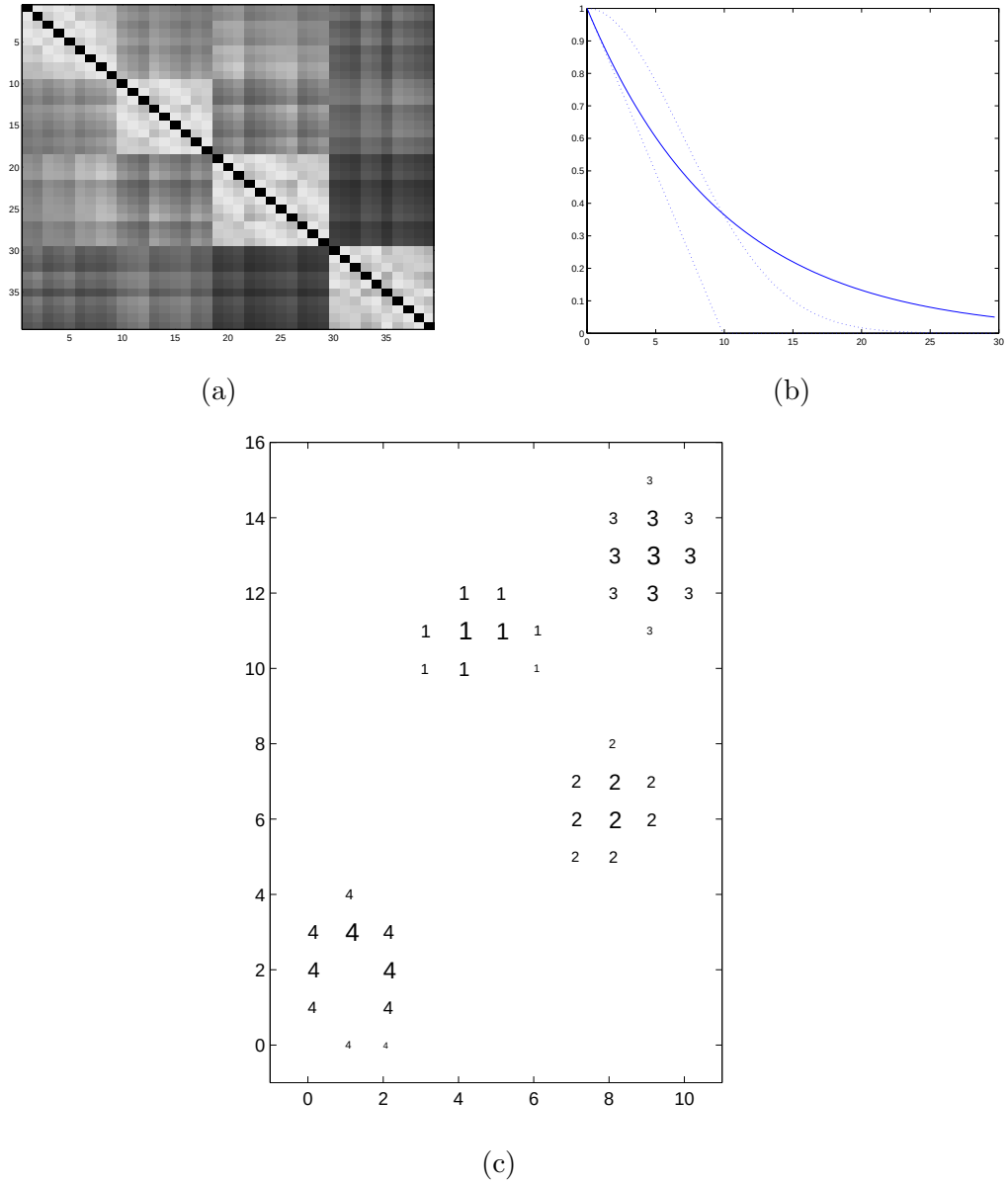


Figure 2.12: Robustness with respect to the weighting function. (a) The similarity matrix a_{ij}. (b) The weighting function $w(i,j) = \exp(-d_{ij}/9.9)$ shown in solid line. The dotted lines show the two alternative weighting functions used in Figures 2.11 and 2.13. (c) Clustering result.

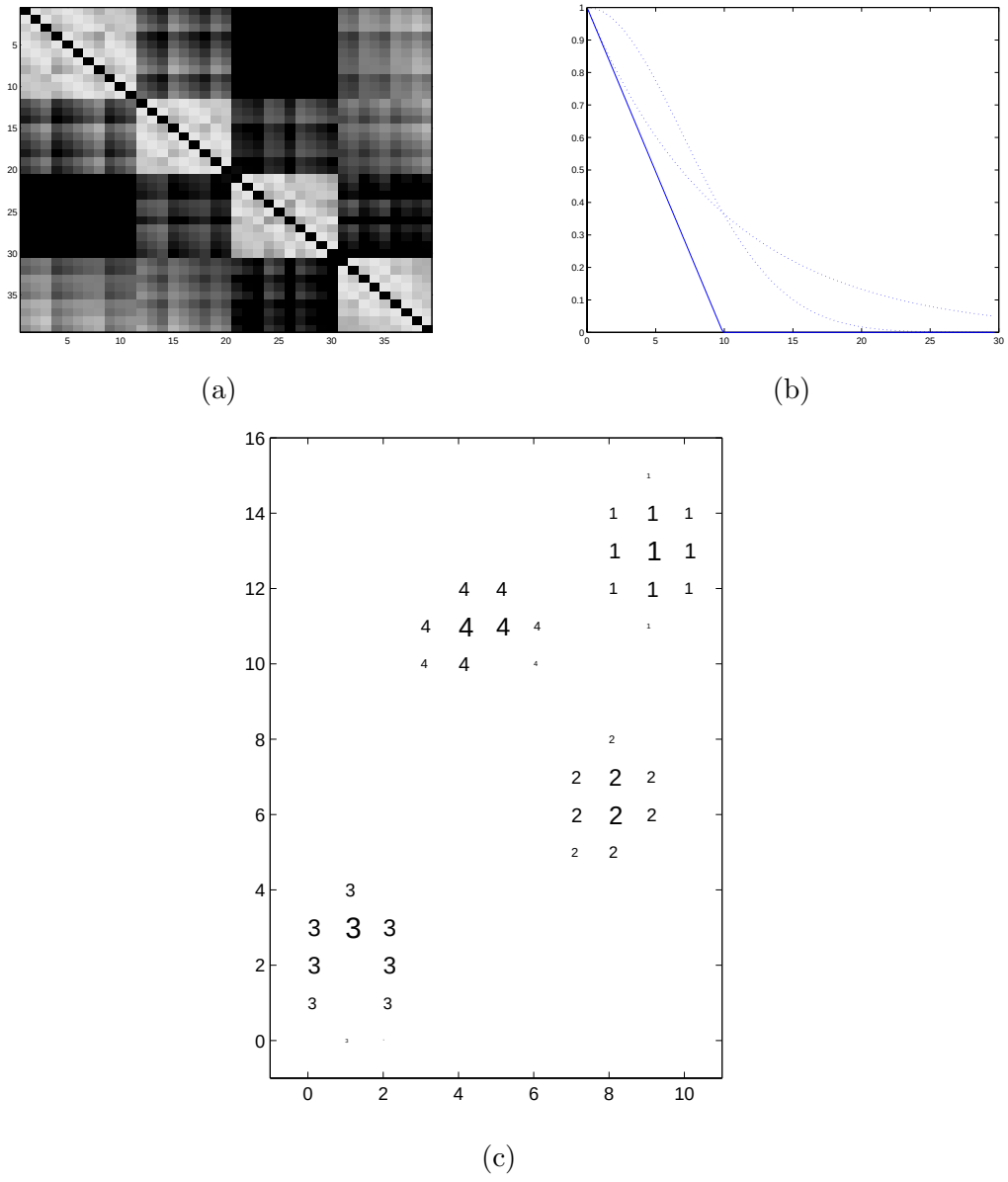
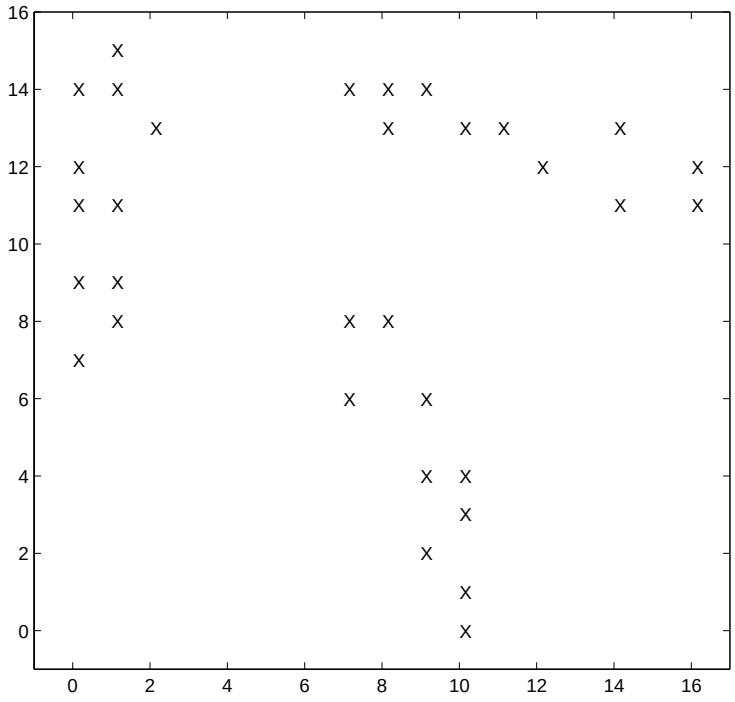
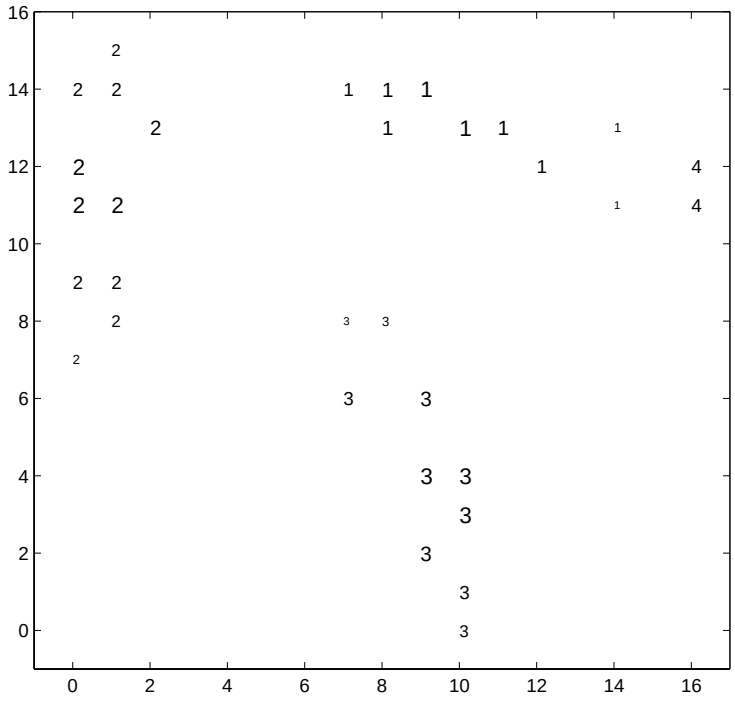


Figure 2.13: Robustness with respect to the weighting function. (a) The similarity matrix a_{ij}. (b) The weighting function $w(i,j) = \max(1 - d_{ij}/9.9, 0)$ shown in solid line. The dotted lines show the two alternative weighting functions used in Figures 2.11 and 2.12. (c) Clustering result.

correctly discovered the globular structure on the center of the figure. On the other hand, it incorrectly split in several parts the outer cluster.

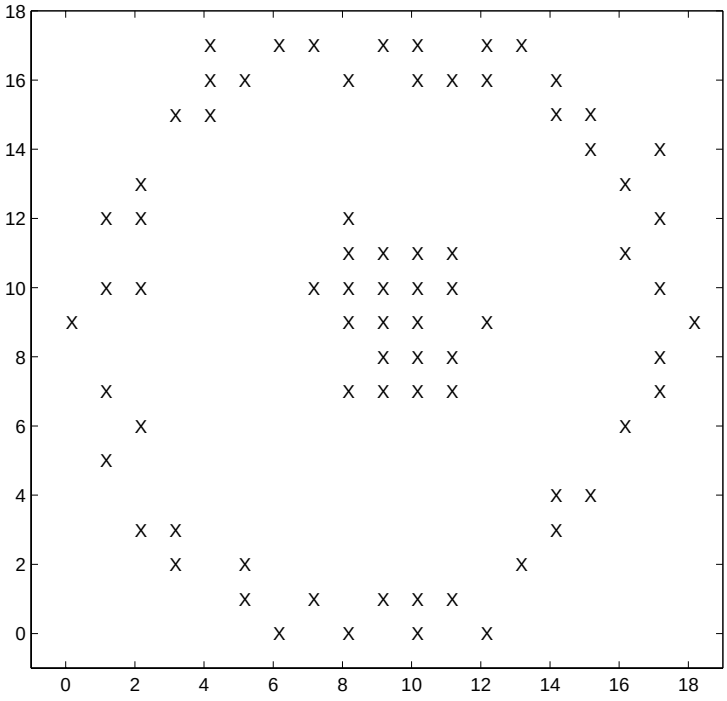


(a)

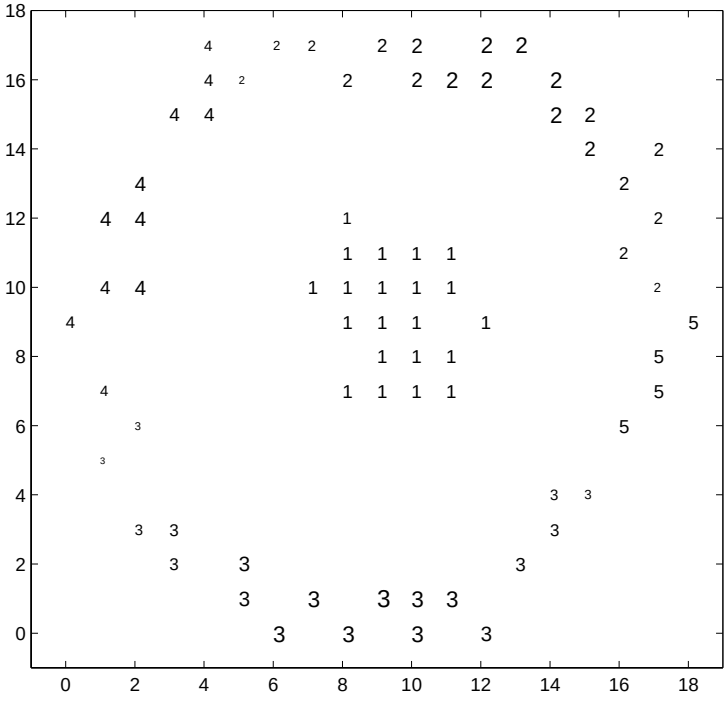


(b)

Figure 2.14: Clustering of points in the Euclidean plane. (a) The 32 data points. (b) Clustering result.



(a)



(b)

Figure 2.15: Clustering of points in the Euclidean plane. (a) The 75 data points. (b) Clustering result.

2.4 Unweighted Graphs

In this section, we turn our attention to unweighted graphs, namely graphs where $w(i,j) = a > 0$ for all $\{i,j\} \in E$, and we show that all the concepts introduced previously do correspond to standard graph-theoretic notions related to that of a clique.

To this end, note that if $C \subseteq V$ is a clique with m vertices of an unweighted graph G, then for any $i \in C$ and $j \notin C$ we have:

$$\phi_C(i,j) = \begin{cases} \frac{a}{m} & \text{if } (i,j) \in E \\ a(\frac{1}{m} - 1) & \text{otherwise.} \end{cases}$$

Given a subset of vertices $S \subseteq V$ and $j \in V$, we denote with:

$$\mathrm{deg}_S(j) = |\{i \in S : \{i,j\} \in E\}| \ ,$$

the degree of j w.r.t. S.

Proposition 2.1 *Let $C \subseteq V$ be a clique with m vertices of an unweighted graph. Then, we have:*

$$\mathrm{w}_{C\cup\{i\}}(i) = a^m(\mathrm{deg}_{C\cup\{i\}}(i) - m + 1) \ ,$$

for any $i \notin C$.

Proof: We proceed by induction on m. If $m = 0$, $w_{\emptyset\cup\{i\}}(i) = 1 = a^0(0 - 0 + 1)$. If $m \geq 1$, we have by Definition 2.1:

$$\mathrm{w}_{C\cup\{i\}}(i) = \sum_{j\in C} \phi_C(j,i)\mathrm{w}_C(j) \ .$$

Applying the inductive hypothesis on $C \setminus \{j\}$, we obtain:

$$\mathrm{w}_C(j) = a^{m-1}(m - 1 - (m-1) + 1) = a^{m-1} \ .$$

Therefore:

$$\begin{aligned} \mathrm{w}_{C\cup\{i\}}(i) &= a^{m-1}\Big(\sum_{j\in C,\ (i,j)\in E} \phi_C(j,i) + \sum_{j\in C,\ (i,j)\notin E} \phi_C(j,i)\Big) \\ &= a^{m-1}\left\{\frac{a}{m}\mathrm{deg}_{C\cup\{i\}}(i) + a\left(\frac{1}{m} - 1\right)\Big[m - \mathrm{deg}_{C\cup\{i\}}(i)\Big]\right\} \\ &= a^m(\mathrm{deg}_{C\cup\{i\}}(i) - m + 1) \ . \end{aligned}$$

□

Note that a clique C of G is maximal if no external vertex to C is connected with more than $|C| - 1$ vertices of C. In addition, we say that C is *strictly* maximal if no vertex external to C is connected with exactly $|C| - 1$ vertices of C. Using these facts and Proposition 2.1, it is simple to prove that, for any clique C with m vertices of G, the following facts hold:

1. $\mathrm{w}_C(i) = a^{m-1}$, for any $i \in C$ and hence $\mathrm{W}(C) = ma^{m-1}$.
2. C is a maximal clique if and only if $\mathrm{w}_{C\cup\{j\}}(j) \leq 0$ for any $j \notin C$.
3. C is a strictly maximal clique if and only if $\mathrm{w}_{C\cup\{j\}}(j) < 0$ for any $j \notin C$.

From the above properties and the fact (see Section 2.1) that $\mathrm{w}_{\{i,j\}}(i) = \mathrm{w}_{\{i,j\}}(j) = a_{ij}$ for all $i, j \in V$ $(i \neq j)$, the next corollary follows.

Corollary 2.1 *Let $S \subseteq V$ be a subset of vertices of an unweighted graph. Then the following statements are equivalent:*

1. *S is a clique.*
2. *$\mathrm{w}_{\{i,j\}}(i) \neq 0$, for all $i, j \in V$ $(i \neq j)$.*
3. *$\mathrm{W}(T) > 0$ for any non-empty subset $T \subseteq S$.*

Moreover, S is a strictly maximal clique if and only if it is dominant.

The fact that our definition of a dominant set is equivalent to that of a (strictly) maximal clique when applied to unweighted graphs is a further motivation to consider dominant sets as clusters.

2.5 Dominant Sets versus Maximal Cliques

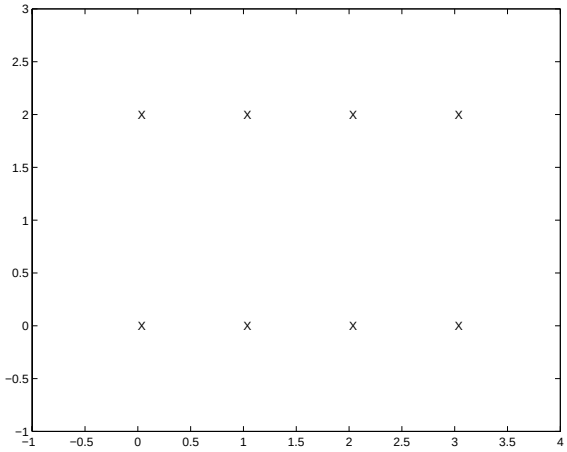
Unweighted similarity graphs provide only binary information about the similarities between data items (either two data items are similar or not), and they are typically derived from weighted similarity graphs by means of some threshold operation [53, 3, 40].

Maximal-clique-based clustering algorithms, which reduce to a search for a maximal clique in the thresholded graph [3, 96], seek compact and isotropic structures. For example,

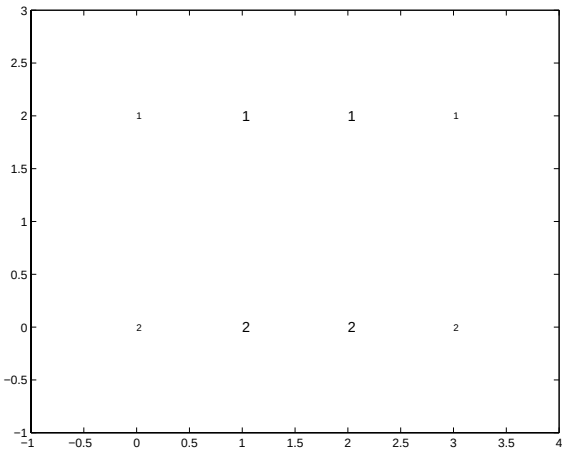
referring to the point set in Figure 2.16, it can be proven that no threshold value may separate the four points on the horizontal top stripe from those on the bottom one. Put another way, it does not exist a threshold value such that the resulting unweighted graph contains exactly two (strictly) maximal cliques corresponding to the two stripes of points.

On the other hand, moving back to the context of edge-weighted graphs, it turns out that, employing the weighting function $w(i,j) = \exp(-d_{ij}^2/2.3^2)$ (see Section 2.2), the two stripes of points correctly form two distinct dominant sets in the (weighted) similarity graph. Interestingly, the point set in Figure 2.17, which is obtained by deleting a point from a stripe of points in Figure 2.16, is not dominant. For example the leftmost point, which is relatively far from the other ones, does not satisfy the condition regarding internal homogeneity of a dominant set (namely, first condition of Definition 2.2). In fact, it turns out that its weight is negative. We conclude that the coherency between the points in Fig. 2.16 is achieved through the central "mediating" item.

Here is the key point: in the classical clique concept, every subset of a clique is still a clique. Put another way, if a set of elements is sufficiently coherent to form a clique, then also every subset is as much coherent. On the other hand, if a set of elements is sufficiently coherent to be dominant, then it is not always true that all its subsets are as much coherent. This is quite different from the clique concept of a cluster, where any pair of items must be completely coherent, regardless of any other mediating items. This is also why our algorithm has a bias for finding compact clusters with some degree of elongation.



(a)



(b)

Figure 2.16: Clustering of points in the Euclidean plane. (a) The 8 data points. (b) Clustering result.

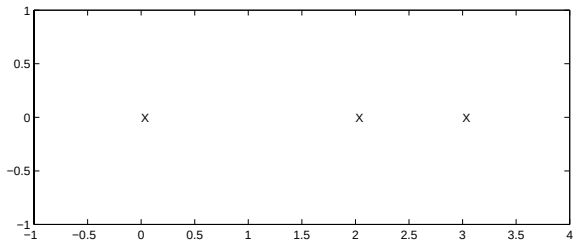


Figure 2.17: A stripe of points from Figure 2.16 with a missing item.

Chapter 3

Clustering as Standard Quadratic Optimization

In this chapter we establish a correspondence between dominant sets and the extrema of a continuous quadratic form over the standard simplex. This will allow us to use continuous optimization techniques to find them. The proposed objective function, which quantifies the degree of cohesiveness of a cluster, has also been recently used by other authors, but using different constraints [104, 91], and our program is a generalization of the so-called Motzkin-Straus problem from graph theory [74].

3.1 From Dominant Sets to Local Optima

Given an edge-weighted graph $G = (V, E, w)$ and its weighted adjacency matrix A, consider the following quadratic program (which is a generalization of the so-called Motzkin-Straus program [74]):

$$\begin{aligned} &\text{maximize} && f(\mathbf{x}) = \tfrac{1}{2}\mathbf{x}^{\mathrm{T}} A \mathbf{x} \\ &\text{subject to} && \mathbf{x} \in \Delta \end{aligned} \tag{3.1}$$

where

$$\Delta = \{\mathbf{x} \in \mathbf{R}^n : \mathbf{x} \geq \mathbf{0} \text{ and } \mathbf{e}^{\mathrm{T}}\mathbf{x} = 1\}$$

is the standard simplex of the n-dimensional Euclidean space $\mathbf{R}^n$ (see Figure 3.1).

Recall that a point $\mathbf{x}^* \in \Delta$ is said to be a *global* solution of program (3.1) if $f(\mathbf{x}^*) \geq f(\mathbf{x})$, for all $\mathbf{x} \in \Delta$. It is said to be a *local* solution if there exists an $\varepsilon > 0$ such that $f(\mathbf{x}^*) \geq f(\mathbf{x})$ for all $\mathbf{x} \in \Delta$ whose distance from $\mathbf{x}^*$ is less than ε, and if $f(\mathbf{x}^*) = f(\mathbf{x})$ implies $\mathbf{x}^* = \mathbf{x}$, then $\mathbf{x}^*$ is said to be a *strict* local solution.

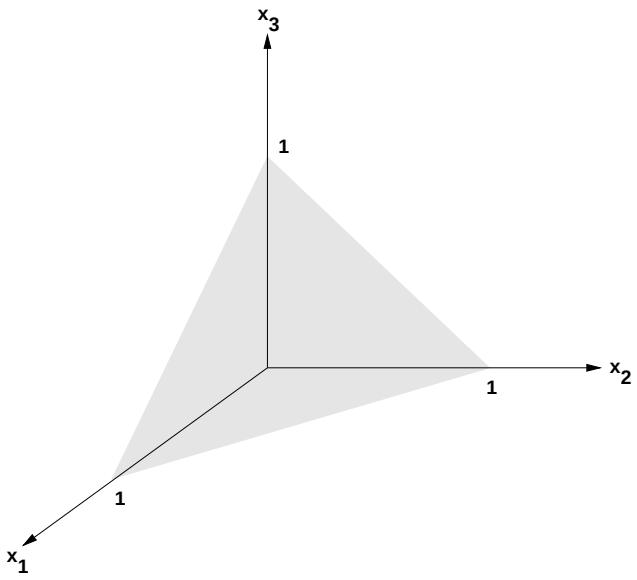


Figure 3.1: The simplex Δ in $\mathbf{R}^3$.

Given a vector $\mathbf{x} \in \mathbf{R}^n$, the *support* of $\mathbf{x}$ is defined as the set of indices corresponding to its non-zero components, that is:

$$\sigma(\mathbf{x}) = \{i \in V \ : \ x_i \neq 0\} \ . \tag{3.2}$$

A point $\mathbf{x} \in \Delta$ satisfies the Karush-Kuhn-Tucker (KKT) conditions for problem (3.1), i.e., the first-order necessary conditions for local optimality [66], if there exist $n+1$ real constants (Lagrange multipliers) $\mu_1, \ldots, \mu_n$ and λ, with $\mu_i \geq 0$ for all $i = 1 \ldots n$, such that:

$$(A\mathbf{x})_i - \lambda + \mu_i = 0 \tag{3.3}$$

for all $i = 1 \ldots n$, and

$$\sum_{i=1}^{n} x_i \mu_i = 0 \ . \tag{3.4}$$

Note that, since both x_i and μ_i are nonnegative for all $i = 1 \ldots n$, the latter condition is equivalent to saying that $i \in \sigma(\mathbf{x})$ implies $\mu_i = 0$. Hence, the KKT conditions can be rewritten as:

$$(A\mathbf{x})_i \begin{cases} = & \lambda, \quad \text{if } i \in \sigma(\mathbf{x}) \\ \leq & \lambda, \quad \text{otherwise} \end{cases} \tag{3.5}$$

for some real constant λ (indeed, it is immediate to see that $\lambda = \mathbf{x}^{\mathrm{T}} A\mathbf{x}$). A point $\mathbf{x} \in \Delta$ satisfying (3.5) will be called a *KKT point* throughout.

We now introduce some new notations. If $S \subseteq V$, we denote by A_S the submatrix of A formed by the rows and the columns indexed by the elements of S. Additionally, we define the matrix B_S as:

$$B_S = \begin{pmatrix} 0 & \mathbf{e}^{\mathrm{T}} \\ \mathbf{e} & A_S \end{pmatrix}$$

where $\mathbf{e}$ is a vector of appropriate length consisting of unit entries. Assuming $S = \{i_1, \ldots, i_m\}$ with $i_1 < \cdots < i_m$, the matrix ${}^j B_S$ is defined to be:

$${}^j B_S = \begin{pmatrix} 0 & & & & \mathbf{e}^{\mathrm{T}} & & \\ \mathbf{e} & A_S^1 & \cdots & A_S^{j-1} & \mathbf{0} & A_S^{j+1} & \cdots & A_S^m \end{pmatrix}$$

where A_S^i denotes the i-th column of A_S.

With these notations, note that the KKT equality conditions in (3.5) amount to saying that there exists a real number λ such that:

$$B_\sigma \left(\lambda, x_{i_1}, \ldots, x_{i_m}\right)^{\mathrm{T}} = (1, 0, \ldots, 0)^{\mathrm{T}} \tag{3.6}$$

where $\sigma = \sigma(\mathbf{x}) = \{i_1, \ldots, i_m\}$ with $i_1 < \cdots < i_m$.

Definition 3.1 *We say that a non-empty subset of vertices S admits* weighted characteristic vector $\mathbf{x}^S \in \Delta$ *if it has non-null total weight* $\mathrm{W}(S)$*, in which case we set:*

$$x_i^S = \begin{cases} \frac{\mathrm{w}_S(i)}{\mathrm{W}(S)}\ , & \text{if } i \in S \\ 0\ , & \text{otherwise.} \end{cases} \tag{3.7}$$

Note that, by definition, dominant sets always admit a weighted characteristic vector.

We now provide a useful characterization of the graph-theoretic notions introduced in Section 2.1 in terms of determinants. Let $S = \{i_1, \ldots, i_m\} \subseteq V$ be a non-empty subset of vertices and, w.l.o.g., assume $i_1 < \cdots < i_m$. Then, we have:

$$\mathrm{w}_S\left(i_h\right) = (-1)^m \det\left({}^h B_S\right), \tag{3.8}$$

for any $i_h \in S$. Moreover:

$$\mathrm{W}(S) = (-1)^m \det\left(B_S\right)\ . \tag{3.9}$$

Conversely, the determinant of the matrix A_S can be characterized in terms of the graph-theoretic notions introduced in Section 2.1:

$$\det(A_S) = (-1)^{m+1} \sum_{i \in S} \mathrm{a}wdeg_S(i) \mathrm{w}_S(i) \ . \tag{3.10}$$

Moreover:

$$\det(A_S) = (-1)^{m+1} \sum_{j \in S} a_{ij} \mathrm{w}_S(j), \tag{3.11}$$

where i is an arbitrary element of S (it can be shown that the sum in (3.11) does not depend upon the choice of i). Finally, we provide an alternative, useful way of computing the $\mathrm{w}_S(i)$'s:

$$\mathrm{w}_S(i) = \sum_{j \in S \setminus \{i\}} (a_{ij} - a_{hj}) \, \mathrm{w}_{S \setminus \{i\}}(j) \tag{3.12}$$

where h is an arbitrary element of $S \setminus \{i\}$ (it can be shown that the sum in (3.12) does not depend upon the choice of h). The proofs of all these properties, which proceed by induction and exploit elementary properties of the determinant, can be found in the Appendix.

The next two results establish useful connections between KKT points of program (3.1) and weighted characteristic vectors.

Lemma 3.1 *Let $\sigma = \sigma(\mathbf{x})$ be the support of a vector $\mathbf{x} \in \Delta$ which admits weighted characteristic vector $\mathbf{x}^\sigma$. Then $\mathbf{x}$ satisfies the KKT equality conditions in (3.5) if and only if $\mathbf{x} = \mathbf{x}^\sigma$. Moreover, in this case, we have:*

1. $$\frac{\mathrm{w}_{\sigma \cup \{j\}}(j)}{\mathrm{W}(\sigma)} = (A\mathbf{x})_j - (A\mathbf{x})_i = -\mu_j \tag{3.13}$$

   for all $i \in \sigma$ and $j \notin \sigma$, where the μ_j's are the (nonnegative) Lagrange multipliers of program (3.1).

2. $$\begin{aligned} f(\mathbf{x}^\sigma) &= \frac{1}{2} \frac{\sum_{i \in \sigma} \mathrm{a}wdeg_\sigma(i) \, \mathrm{w}_\sigma(i)}{\mathrm{W}(\sigma)} \\ &= \frac{1}{2} \sum_{j \in \sigma} a_{ij} \frac{\mathrm{w}_\sigma(j)}{\mathrm{W}(\sigma)} \ , \end{aligned}$$

   where, in the last equality, i is an arbitrary element of σ.

Proof: Note that conditions (3.6), which are equivalent to the KKT equality conditions in (3.5), can be regarded as a system of linear equations in the unknowns λ and x_i's ($i \in \sigma$). From Lemma A.1, the system has a unique solution since $\det(B_\sigma) \neq 0$. Hence, supposing $\sigma = \{i_1, \ldots, i_m\}$ and, w.l.o.g., $i_1 < \ldots < i_m$, from Cramer's rule and Lemma A.1 we have:

$$x_{i_h} = \frac{\det\left({}^{h}B_\sigma\right)}{\det(B_\sigma)} = \frac{(-1)^m \, \mathrm{w}_\sigma(i_h)}{(-1)^m \, \mathrm{W}(\sigma)} = \frac{\mathrm{w}_\sigma(i_h)}{\mathrm{W}(\sigma)}$$

for any $1 \leq h \leq m$. Therefore $\mathbf{x} = \mathbf{x}^\sigma$.

The fact that $(A\mathbf{x})_j - (A\mathbf{x})_i = -\mu_j$, for $i \in \sigma$ and $j \notin \sigma$, follows immediately from equation (3.3). Finally, using equation (3.12), we obtain:

$$\begin{aligned} \frac{\mathrm{w}_{\sigma\cup\{j\}}(j)}{\mathrm{W}(\sigma)} &= \frac{\sum_{h\in\sigma}(a_{jh} - a_{ih})\mathrm{w}_\sigma(h)}{\mathrm{W}(\sigma)} \\ &= \sum_{h\in\sigma} a_{jh}x_h^\sigma - \sum_{h\in\sigma} a_{ih}x_h^\sigma \\ &= (A\mathbf{x}^\sigma)_j - (A\mathbf{x}^\sigma)_i \end{aligned}$$

which concludes the proof of 1, since $\mathbf{x} = \mathbf{x}^\sigma$.

Since $\mathbf{x}^\sigma$ satisfies the KKT equality conditions in (3.5), it is immediate to see that, for any $i \in \sigma$:

$$f(\mathbf{x}^\sigma) = \frac{1}{2}(A\mathbf{x}^\sigma)_i = \frac{1}{2}\sum_{j\in\sigma} a_{ij}\frac{\mathrm{w}_\sigma(j)}{\mathrm{W}(\sigma)}$$

The fact that $f(\mathbf{x}^\sigma) = \frac{1}{2}\frac{\sum_{i\in\sigma}\mathrm{awdeg}_\sigma(i)\mathrm{w}_\sigma(i)}{\mathrm{W}(\sigma)}$ follows easily from equations (3.10) and (3.11).

□

Proposition 3.1 *Let $\mathbf{x} \in \Delta$ be a vector whose support $\sigma = \sigma(\mathbf{x})$ has positive total weight $\mathrm{W}(\sigma)$, and hence admitting weighted characteristic vector $\mathbf{x}^\sigma$. Then, $\mathbf{x}$ is a KKT point for (3.1) if and only if the following conditions hold:*

1. $\mathbf{x} = \mathbf{x}^\sigma$
2. $\mathrm{w}_{\sigma\cup\{j\}}(j) \leq 0$, *for all* $j \notin \sigma$.

Proof: Vector $\mathbf{x}$ satisfies the KKT conditions (3.5) if and only if $\mathbf{x} = \mathbf{x}^\sigma$ (cfr. Lemma 3.1) and $(A\mathbf{x})_j \leq (A\mathbf{x})_i$ for any $j \notin \sigma$ and $i \in \sigma$, but from (3.13) the latter condition amounts to saying that $\mathrm{w}_{\sigma\cup\{j\}}(j) \leq 0$, since $\mathrm{W}(\sigma) > 0$. □

The following theorem, which is the main result of this chapter, establishes an intriguing connection between dominant sets and local solutions of program (3.1).

Theorem 3.1 *If S is a dominant subset of vertices, then its weighted characteristics vector $\mathbf{x}^S$ is a strict local solution of program (3.1).*

Conversely, if $\mathbf{x}^*$ is a strict local solution of program (3.1) then its support $\sigma = \sigma(\mathbf{x}^*)$ is a dominant set, provided that $\mathrm{w}_{\sigma\cup\{i\}}(i) \neq 0$ for all $i \notin \sigma$.

Proof:

First we note that, using equation (3.9), the well-known bordered Hessian test from nonlinear programming [66] can easily be reformulated in the following way: Given a subset of m vertices $Q \subseteq V$, A_Q is negative definite in the subspace $\{\mathbf{y} \in \mathbf{R}^m : \sum_{i=1}^m y_i = 0\}$ if and only if $\mathrm{W}(T) > 0$ for any non-empty subset $T \subseteq Q$.

Now, let S be a dominant set. Then, from Proposition 3.1, it follows that $\mathbf{x}^S$ is a KKT point for (3.1). Moreover, by Lemma 3.1, we have that the j-th nonnegative Lagrange multiplier μ_j ($j \notin S$) is positive if and only if $\mathrm{w}_{S\cup\{j\}}(j) < 0$. Therefore, the second-order sufficient conditions for local optimality [66], together with the bordered Hessian test, imply that $\mathbf{x}^S$ is a strict local solution for program (3.1).

Conversely, suppose that $\mathbf{x}^*$ is a strict local solution of (3.1), and let $\sigma = \sigma(\mathbf{x}^*)$ be its support. After some algebra, it follows that the submatrix A_σ is negative definite in the subspace $\{\mathbf{y} \in \mathbf{R}^m : \sum_{i=1}^m y_i = 0\}$, where $m = |\sigma|$. Hence, from the bordered Hessian test, we have $\mathrm{W}(T) > 0$ for any non-empty subset $T \subseteq \sigma$.

Moreover, we have $\mathrm{w}_S(i) > 0$ for all $i \in S$. This follows directly from Lemma 3.1 (in fact, $\mathbf{x}^*$ is a KKT point) and the definition of weighted characteristic vector. Finally, Proposition 3.1 states that $\mathbf{x}^* = \mathbf{x}^\sigma$ and $\mathrm{w}_{\sigma\cup\{j\}}(j) \leq 0$, for all $j \notin \sigma$. Therefore, the fact that σ is dominant follows trivially from the hypotheses. □

The condition that $\mathrm{w}_{\sigma\cup\{i\}}(i) \neq 0$ for all $i \notin \sigma$ is a technicality due to the presence of "spurious" solutions in (3.1), namely solutions whose support does not admit a weighted characteristic vector. However, this corresponds to a non-generic situation and thus, in the following, we shall ignore it.

By virtue of Theorem 3.1 dominant sets are in correspondence with (strict) solutions of quadratic program (3.1). This is interesting because recently other quadratic programming formulations have been proposed for clustering and segmentation, though motivated by the different idea of finding cuts in a similarity graph [112, 44] or computing eigenvalues and eigenvectors of the weighted adjacency matrix [91, 104]. Once the clustering problem is formulated as a continuous optimization problem, we can use any optimization technique to solve it. In the next chapter, we shall see that payoff-monotonic dynamics from evolutionary game theory lend themselves well to this task.

Relation to Sarkar and Boyer's Approach

Our continuous quadratic formulation of the clustering problem can be also motivated as follows. A common way to represent a cluster of nodes (see, e.g., [104, 91, 112]) is to associate a real n-dimensional vector ($\mathbf{x}$) to it, where its components express the participation of nodes in the cluster: if a component has a small value, then the corresponding node is weakly associated with the cluster, whereas if it has a large value, the node is strongly associated with the cluster. Components corresponding to nodes not participating in the cluster are zero.

Intuitively, a good cluster is one where elements that are strongly associated with it also have large values connecting one another in the similarity matrix. Hence, a natural way of defining the cohesiveness of a cluster is given by the function $\mathbf{x}^{\mathrm{T}} A\mathbf{x}$ and this allows us to formulate the clustering problem as the problem of finding a vector $\mathbf{x}$ that maximizes f. However, note that the objective function is useless without some normalization of the components of $\mathbf{x}$, and thus we impose to it simplex (or probability) constraints. This yields the standard quadratic program (3.1). Thus, in our formulation, a maximally cohesive cluster can be found by determining a solution of program (3.1).

A similar (though substantially different) formulation to pairwise clustering has recently been proposed by Sarkar and Boyer [104] (see also [91]), who suggested maximizing $\mathbf{x}^{\mathrm{T}} A\mathbf{x}$ subject to $\mathbf{x}^{\mathrm{T}}\mathbf{x} = 1$, leading to a continuous quadratic optimization problem over the sphere instead of the simplex. Interestingly, this is equivalent to determining the eigenvalues (and corresponding eigenvectors) of the similarity matrix A, a problem for which standard algorithms exist. However, this "spectral" approach has a serious drawback, in that in general the components of an eigenvector are not necessarily nonnegative.

Eigenvectors with negative components are meaningless in the above interpretation since the membership of a node in a cluster cannot be negatively valued. This lead Sarkar and Boyer, somewhat arbitrarily, to consider only same-sign eigenvectors, thereby ignoring the others. In contrast to Sarkar and Boyer's approach, we automatically avoid the nuisance of dealing with negative components. As an aside, we also note that no exact combinatorial interpretation is offered for their "eigenclusters."

Relation to the Motzkin-Straus Theorem

Before concluding this chapter, we note that program (3.1) includes as a special case the one introduced by Motzkin and Straus [74], which is intimately related to the maximum clique problem in the context of unweighted graphs.

The problem of determining a maximum clique of an unweighted graph (or finding its clique number) is a classical combinatorial optimization problem with a variety of practical applications. It is known to be *NP*-hard for arbitrary graphs and, according to recent theoretical results, so is the problem of approximating it within a constant factor. We refer to [12] for a recent review concerning algorithms, applications and complexity issues of this important problem.

In 1965, Motzkin and Straus [74] established a remarkable connection between the solutions of program (3.1), where A is the (unweighted[1]) adjacency matrix of a graph G, and $\omega(G)$ denotes its clique number. Specifically, if $\mathbf{x}^*$ is a global solution of (3.1), they showed that the clique number $\omega(G)$ of G is related to $f(\mathbf{x}^*)$ according to the following formula:

$$\omega(G) = \frac{1}{1 - 2f(\mathbf{x}^*)} \ . \tag{3.14}$$

Additionally, they proved that a subset of vertices S is a maximum clique of G if and only if its (unweighted) *characteristic vector* $\mathbf{x}^{S,\mathrm{u}}$, which is the vector in Δ defined as:

$$x_i^{S,\mathrm{u}} = \begin{cases} 1/|S|, & \text{if } i \in S \\ 0, & \text{otherwise} \end{cases}$$

is a global solution of program (3.1).[2]

[1] The original setting of Motzkin and Straus corresponds to unweighted graphs where $w(i,j) = 1$ for all $\{i,j\} \in E$.

[2] Actually, in their original paper, Motzkin and Straus proved just the "only-if" part of this theorem. The converse direction is however a straightforward consequence of their result [88].

Recently, a similar connection has been established between *maximal* cliques of G and local solutions of program (3.1), [88, 37]. Therefore, it is not surprising that, in the unweighted case, dominant sets turn out to be equivalent to (strictly) maximal cliques.

The connection established by the Motzkin-Straus theorem between a purely combinatorial problem and a continuous one, although probably motivated by genuine mathematical curiosity, is indeed a fruitful one both from a theoretical and a practical standpoint. For example, in their original paper Motzkin and Straus used their result to provide an alternative proof of Turán's classical theorem, and Wilf [119] derived from it new bounds on the clique size. Computationally, the result has also proven to be extremely useful since it motivated several clique-finding heuristics (see, e.g., [12] and references therein). Moreover, the Motzkin-Straus theorem has recently found various applications in computer vision and pattern recognition, especially in graph matching and related problems [86, 89, 87].

During the last few years, several generalizations of the Motzkin-Straus theorem have been proposed including, for example, Bomze's spurious-free regularized version [11], Gibbons et al.'s extension to vertex-weighted graphs [37], based on a suggestion by Lovász, and Sós and Straus' generalization to hypergraphs [114]. Our combinatorial characterization of the solutions of program (3.1) can be also thought of a generalization to edge-weighted graphs of the original Motzkin-Straus theorem.

Chapter 4

Clustering Using Monotone Game Dynamics

Our continuous quadratic formulation of the clustering problem allows us to tackle it by means of dynamic equations from evolutionary game theory, a discipline pioneered by J. Maynard Smith [72] that aims to model the evolution of animal behavior using the principles and tools of noncooperative game theory [48]. Evolutionary game models are also gaining increasing popularity in economics since they allow one to elegantly get rid of the much-debated assumptions of traditional game theory concerning the full rationality and complete knowledge of players [117, 103, 28]. In particular, we focus on a wide family of evolutionary game dynamics characterized by having the growth rates of strategies ordered by their expected payoffs, so that strategies associated to higher payoffs grow faster. These *payoff-monotonic* dynamics, which have already proven to be effective in graph matching and related problems [86, 87, 93], are attractive as can be coded in a few lines of any high-level programming language, can easily be implemented in a parallel network of locally interacting units, and offer the advantage of biological plausibility [123]. We show that, when the payoff matrix is symmetric, these models possess a quadratic Liapunov function which is strictly increasing along any nonconstant trajectory, and their asymptotically stable stationary points are in one-to-one correspondence with (strict) local solutions of standard quadratic programs. These properties make therefore *any* member of the payoff-monotonic family a potential heuristic for our pairwise data clustering problem.

We start in Section 4.1 with background on payoff-monotonic game dynamics. The connections between this family of dynamics and quadratic optimization problems are outlined in Section 4.2, where we also show that payoff-monotonic dynamics can be usefully employed to find data clusters. Finally, in Section 4.3 we expose the dynamics we used in

our experiments.

4.1 Payoff-Monotonic Game Dynamics

Evolutionary game theory considers an idealized scenario whereby in a large population pairs of individuals are repeatedly drawn at random to play a symmetric two-player game. In contrast to traditional game theoretic models, players are not supposed to behave rationally or to have complete knowledge of the details of the game. They act instead according to a pre-programmed behavior pattern, or pure strategy, and it is supposed that some evolutionary selection process operates over time on the distribution of behaviors. We refer the reader to [48, 117] for excellent introductions to this rapidly expanding field.

Let $J = \{1, \cdots, n\}$ be the set of available pure strategies and, for all $i \in J$, let $x_i(t)$ be the proportion of population members playing strategy i, at time t. The state of the population at a given instant is the vector $\mathbf{x} = (x_1, \cdots, x_n)^{\mathrm{T}}$. Clearly, population states are constrained to lie in the standard simplex Δ of $\mathbb{R}^n$. The support $\sigma(\mathbf{x})$ of a population state $\mathbf{x} \in \Delta$ (as defined in Section 3.1) corresponds to the set of non-extinct strategies. Given a subset of strategies $S \subseteq J$, the set of states where all strategies outside S are extinct, which corresponds to a face of Δ, is defined as:

$$\Delta_S = \{\mathbf{x} \in \Delta \ : \ \sigma(\mathbf{x}) \subseteq S\}$$

and its (relative) interior is:

$$\mathrm{int}(\Delta_S) = \{\mathbf{x} \in \Delta \ : \ \sigma(\mathbf{x}) = S\} \ .$$

Clearly, $\Delta_J = \Delta$ and, accordingly, we shall write $\mathrm{int}(\Delta)$ instead of $\mathrm{int}(\Delta_J)$.

Let $M = (m_{ij})$ be the $n \times n$ payoff or utility matrix. Specifically, for each pair of strategies $i, j \in J$, m_{ij} represents the payoff of an individual playing strategy i against an opponent playing strategy j. In biological contexts payoffs are typically measured in terms of Darwinian fitness or reproductive success (i.e., the player's expected number of offspring), whereas in economic applications they usually represent firms' profits or consumers' utilities. If the population is in state $\mathbf{x}$, the expected payoff earnt by an i-strategist is:

$$\pi_i(\mathbf{x}) = \sum_{j=1}^{n} m_{ij} x_j = (M\mathbf{x})_i \tag{4.1}$$

while the mean payoff over the entire population is

$$\pi(\mathbf{x}) = \sum_{i=1}^{n} x_i \pi_i(\mathbf{x}) = \mathbf{x}^{\mathrm{T}} M \mathbf{x} \ . \tag{4.2}$$

In evolutionary game theory the assumption is made that the game is played over and over, generation after generation, and that the action of natural selection will result in the evolution of the fittest strategies. If successive generations blend into each other, the evolution of behavioral phenotypes can be described by a set of ordinary differential equations. A general class of evolution equations is given by:

$$\dot{x}_i = x_i g_i(\mathbf{x}) \tag{4.3}$$

where a dot signifies derivative with respect to time, and $g = (g_1, \ldots, g_n)$ is a function with open domain containing Δ. Here, the function g_i $(i \in j)$ specifies the rate at which pure strategy i replicates when the population is in state $\mathbf{x}$. It is usually required that the growth function g is *regular* [117], which means that it is C^1 and that $g(\mathbf{x})$ is always orthogonal to $\mathbf{x}$, i.e., $g(\mathbf{x})^{\mathrm{T}}\mathbf{x} = 0$. The former condition guarantees us that the system of differential equations (4.3) has a unique solution through any initial population state. The condition $g(\mathbf{x})^{\mathrm{T}}\mathbf{x} = 0$, instead, ensures that the simplex Δ is invariant under (4.3), namely any trajectory starting in Δ will remain in Δ.

A point $\mathbf{x}$ is said to be a *stationary* (or equilibrium) point of our dynamical system, if $\dot{x}_i = 0$, for all $i = 1 \ldots n$. A stationary point $\mathbf{x}$ is said to be *Liapunov stable* (or, more simply *stable*) if for any neighborhood U of $\mathbf{x}$ there exists a neighborhood W of $\mathbf{x}$ such that any trajectory in W remains also in U (formally, $\mathbf{x}(0) \in W$ implies $\mathbf{x}(t) \in U$ for all $t \geq 0$). It is said to be *asymptotically stable* if, in addition, such trajectories converge to $\mathbf{x}$.

Payoff-monotonic game dynamics represent a wide class of regular selection dynamics for which useful properties hold. Intuitively, for a payoff-monotonic dynamics the strategies associated to higher payoffs will increase at higher rate. Formally, a regular selection dynamics (4.3) is said to be payoff-monotonic if:

$$g_i(\mathbf{x}) > g_j(\mathbf{x}) \iff \pi_i(\mathbf{x}) > \pi_j(\mathbf{x}) \tag{4.4}$$

for all $\mathbf{x} \in \Delta$.

Although this class contains many different dynamics, it turns out that they share many common properties. To begin with, they all have the same set of stationary points.

Proposition 4.1 *A point* $\mathbf{x} \in \Delta$ *is stationary under any payoff-monotonic dynamics if and only if* $\pi_i(\mathbf{x}) = \pi(\mathbf{x})$ *for all* $i \in \sigma(\mathbf{x})$.

Proof. See [117, Proposition 4.7]. □

A well-known subclass of payoff-monotonic game dynamics is given by:

$$\dot{x}_i = x_i \left(\phi(\pi_i(\mathbf{x})) - \sum_{j=1}^{n} x_j \phi(\pi_j(\mathbf{x})) \right) \tag{4.5}$$

where $\phi(u)$ is an increasing function of u. These models arise in modeling the evolution of behavior by way of imitation processes, where players are occasionally given the opportunity to change their own strategies [47, 117].

When ϕ is the identity function, i.e., $\phi(u) = u$, we obtain the standard first-order *replicator equations* [117, 48]:

$$\dot{x}_i = x_i \left(\pi_i(\mathbf{x}) - \sum_{j=1}^{n} x_j \pi_j(\mathbf{x}) \right) \tag{4.6}$$

whose basic idea is that the average rate of increase $\dot{x}_i/x_i$ equals the difference between the average fitness of strategy i and the mean fitness over the entire population. Such dynamics can be implemented in analog VLSI hardware [115].

Another popular model arises when $\phi(u) = e^{\kappa u}$ which yields:

$$\dot{x}_i = x_i \left(e^{\kappa \pi_i(\mathbf{x})} - \sum_{j=1}^{n} x_j e^{\kappa \pi_j(\mathbf{x})} \right) \tag{4.7}$$

where κ is a positive constant. As κ tends to 0, the orbits of this dynamics approach those of the standard, first-order replicator model (4.6), slowed down by the factor κ; moreover, for large values of κ the model approximates the so-called "best-reply" dynamics [47, 48].

4.2 Payoff-Monotonic Dynamics and Quadratic Programming

In this section we shall explore the connections between payoff-monotonic dynamics and quadratic optimization problems. Consider the following standard quadratic program:

$$\begin{array}{ll} \text{maximize} & \pi(\mathbf{x}) = \mathbf{x}^{\mathrm{T}} M \mathbf{x} \\ \text{subject to} & \mathbf{x} \in \Delta \end{array} \tag{4.8}$$

where M is an arbitrary $n \times n$ symmetric matrix. In evolutionary game theory a symmetric payoff matrix arises in the context of doubly symmetric (or partnership) games, where the interests of the two players coincide [48, 117].

Recall from Section 3.1 that a point $\mathbf{x} \in \Delta$ satisfies the KKT conditions for problem (4.8) [1] if there exist $n+1$ real constants $\mu_1, \ldots, \mu_n$ and λ, with $\mu_i \geq 0$ for all $i = 1 \ldots n$, such that:

$$(M\mathbf{x})_i - \lambda + \mu_i = 0$$

for all $i = 1 \ldots n$, and

$$\sum_{i=1}^{n} x_i \mu_i = 0 \ .$$

Since both x_i and μ_i are nonnegative for all $i = 1 \ldots n$, the latter condition is equivalent to saying that $i \in \sigma(\mathbf{x})$ implies $\mu_i = 0$. Hence, the KKT conditions can be rewritten as:

$$(M\mathbf{x})_i \begin{cases} = & \lambda \quad \text{if } i \in \sigma(\mathbf{x}) \\ \leq & \lambda \quad \text{if } i \notin \sigma(\mathbf{x}) \end{cases} \tag{4.9}$$

for some real constant λ. On the other hand, it is immediate to see that $\lambda = \mathbf{x}^{\mathrm{T}} M \mathbf{x}$.

The following easily proved results establishes a first connection between standard quadratic programs and payoff-monotonic dynamics.

Proposition 4.2 *If $\mathbf{x} \in \Delta$ is a KKT point for (4.8), then it is a stationary point of any payoff-monotonic dynamics. If $\mathbf{x} \in \mathrm{int}(\Delta)$ then the converse also holds.*

Proof. A straightforward consequence of Proposition 4.1 and the KKT conditions (4.9). □

In an unpublished paper, Hofbauer [47] showed that for symmetric payoff matrices the population mean payoff $\mathbf{x}^{\mathrm{T}} M \mathbf{x}$ is strictly increasing along the trajectories of any payoff-monotonic dynamics. This result generalizes the celebrated "fundamental theorem of natural selection" [48, 117].

Theorem 4.1 *If the payoff matrix M is symmetric, then $\pi(\mathbf{x}) = \mathbf{x}^{\mathrm{T}} M \mathbf{x}$ is strictly increasing along any nonconstant trajectory of any payoff-monotonic dynamics. In other words, $\dot{\pi}(\mathbf{x}(t)) \geq 0$ for all t, with equality if and only if $\mathbf{x} = \mathbf{x}(t)$ is a stationary point.*

[1] Here, compared with problem (3.1), we drop the multiplicative constant $\frac{1}{2}$. Clearly, the maximizers remain the same.

Proof. See [83]. □

Apart from the monotonicity result which provides a (strict) Liapunov function for payoff-monotonic dynamics, the previous theorem also rules out complicated attractors like cycles, invariant tori, or even strange attractors. It also allows us to establish a strong connection between the stability properties of these dynamics and the solutions of (4.8). To this end, we need an auxiliary result.

Lemma 4.1 *Let* $\mathbf{x}$ *be a strict local solution of (4.8), and put* $S = \sigma(\mathbf{x})$*. Then,* $\mathbf{x}$ *is the only stationary point of any payoff-monotonic dynamics in* $\mathrm{int}(\Delta_S)$*. Moreover,* $\mathbf{x}^{\mathrm{T}} M \mathbf{x} > \mathbf{y}^{\mathrm{T}} M \mathbf{y}$ *for all* $\mathbf{y} \in \Delta_S$*.*

Proof. See [83]. □

Theorem 4.2 *A point* $\mathbf{x} \in \Delta$ *is a strict local solution of program (4.8) if and only if it is asymptotically stable under any payoff-monotonic dynamics.*

Proof. See [83]. □

The results presented in this section show that payoff-monotonic dynamics can be usefully employed to find (local) solutions of standard quadratic programs and hence (according to our formulation) data clusters. Indeed, let $G = (V, E, w)$ be a similarity graph and let A denote its similarity matrix. By using

$$M = A \tag{4.10}$$

as the payoff matrix, any payoff-monotonic dynamics, starting from an arbitrary initial state, will eventually be attracted with probability 1 by the nearest asymptotically stable point which. By virtue of Theorem 4.2, this will be a strict local solution of program (3.1), and hence it will induce a cluster of vertices.

4.3 Finding Dominant Sets by Replicator Dynamics

In all the experiments reported in this thesis we used the following discrete-time model:

$$x_i(t+1) = \frac{x_i(t)\pi_i(t)}{\sum_{j=1}^{n} x_j(t)\pi_j(t)} \tag{4.11}$$

which corresponds to a well-known discretization of (4.6), [117], and is called discrete-time first-order replicator equations. Equations (4.11) turn out to be a special case of a general class of dynamical systems introduced by Baum and Eagon [6] in the context of Markov chains theory. They also represent an instance of the original heuristic Rosenfeld-Hummel-Zucker relaxation labeling algorithm [101], whose dynamical properties have recently been clarified [85] (specifically, it corresponds to the 1-object, n-label case). Moreover, the biological significance of such processes has been strongly advocated by Zucker et al. [123] who hypothesized that the first 2-3 iterations of the algorithm could be implemented by the pyramidal neurons connecting the striate and the extrastriate cortices.

We are now interested in the dynamical properties of discrete-time first-order replicator dynamics; it is these properties that will allow us to solve our original combinatorial problem. Indeed, the discrete-time model enjoys many of the properties of continuous-time replicator equations (4.6). To begin with, it is readily seen that the simplex Δ is invariant under discrete-time replicator equations (4.11). Moreover, the stationary points of (4.11), i.e. the points satisfying $x_i(t+1) = x_i(t)$, and the stationary points of (4.6), coincide and are the solutions of the equations

$$x_i[(M\mathbf{x})_i - \mathbf{x}^{\mathrm{T}} M\mathbf{x}] = 0 \tag{4.12}$$

for $i = 1 \ldots n$.

Theorem 4.3 *If $M = M^{\mathrm{T}}$ then the function $\mathbf{x}(t)^{\mathrm{T}} M\mathbf{x}(t)$ is strictly increasing with increasing t along any non-stationary trajectory $\mathbf{x}(t)$ under discrete-time (4.11) replicator dynamics. Furthermore, any such trajectory converges to a stationary point. Finally, a vector $\mathbf{x} \in \Delta$ is asymptotically stable under (4.11) if and only if $\mathbf{x}$ is a strict local maximizer of $\mathbf{x}^{\mathrm{T}} M\mathbf{x}$ on Δ.*

The previous result is known in mathematical biology as the fundamental theorem of natural selection [48, 117] and, in its original form, traces back to R. A. Fisher [25]. It can

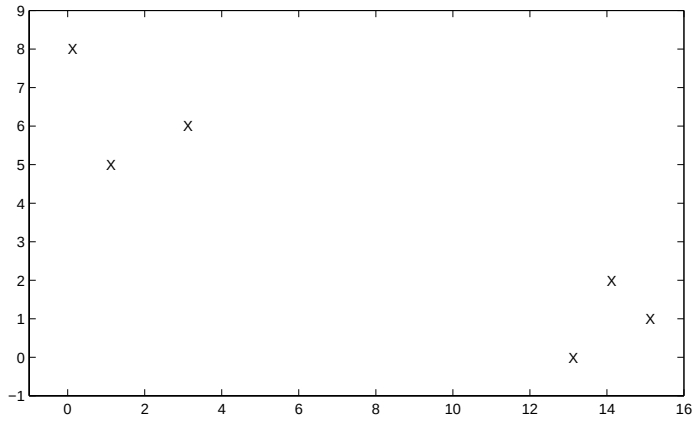


Figure 4.1: Simple point set (6 data points).

be regarded as a straightforward implication of the more general Baum-Eagon theorem [6]. The fact that all trajectories of the replicator dynamics converge to a stationary point has been proven more recently [65, 69].

In light of their dynamical properties, first-order discrete-time replicator equations naturally suggest themselves as a simple and useful heuristic for finding dominant sets. Indeed, let A denote the weighted adjacency matrix of an edge-weighted graph G. By letting

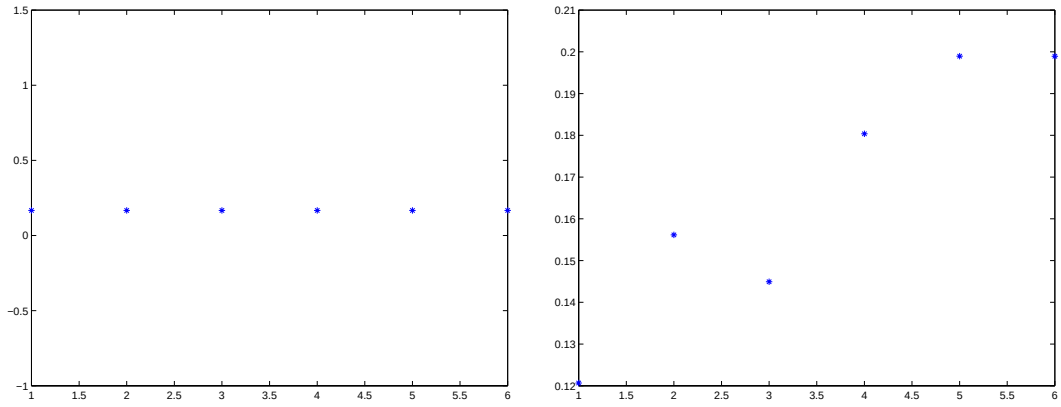
$$M = A$$

we know that the replicator dynamical systems (4.11), starting from an arbitrary initial state, will iteratively maximize the function $\mathbf{x}^{\mathrm{T}} M \mathbf{x}$ over Δ and will eventually be attracted with probability 1 by the nearest asymptotically stable point. By virtue of Theorem 4.3 this will then correspond to a strict local maximizer of $\mathbf{x}^{\mathrm{T}} M \mathbf{x}$ in Δ and hence, by Theorem 3.1, to a dominant set.

Since the process cannot leave the boundary of Δ, it is customary to start out the relaxation process from some interior point, a common choice being the barycenter of Δ. This prevents the search from being initially biased in favor of any particular solution. Getting back to the pseudo-code of Figure 2.1, the procedure DOMINANT_SET simply involves the simulation of the dynamical system (4.11).

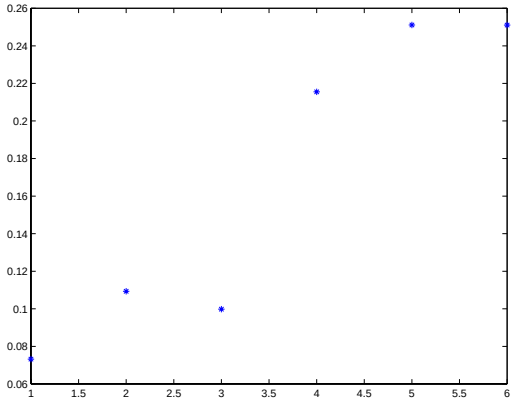
To illustrate, in Figure 4.2 the behavior of the dynamics (4.11) in clustering the simple point set in Figure 4.1 is shown. The Euclidean distances d_{ij} are transformed to similarity weights by using the functional form $w(i,j) = \exp(-d_{ij}^2/3^2)$ (see Chapter 2). The cluster found by the dynamics contains the three rightmost points.

Characterizing the complexity of our approach appears to be difficult since it involves the simulation of a dynamical system. However, we have observed experimentally that the system converges quickly when applied to image segmentation (Chapter 5) as well as image database organization (Chapter 7) and this confirms earlier findings, [11, 86].

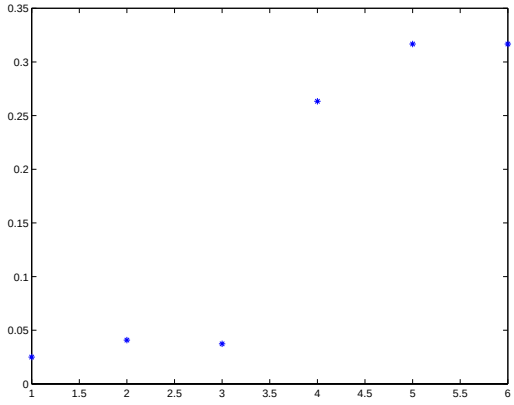


Starting point

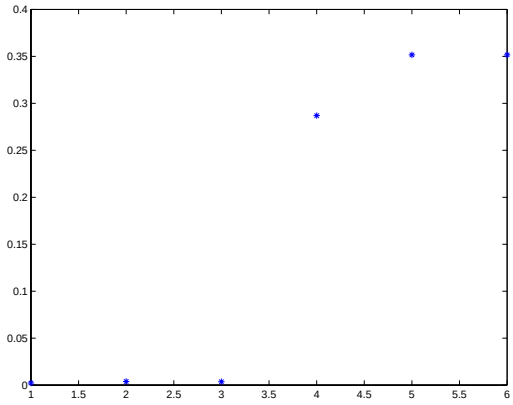
Iteration 1



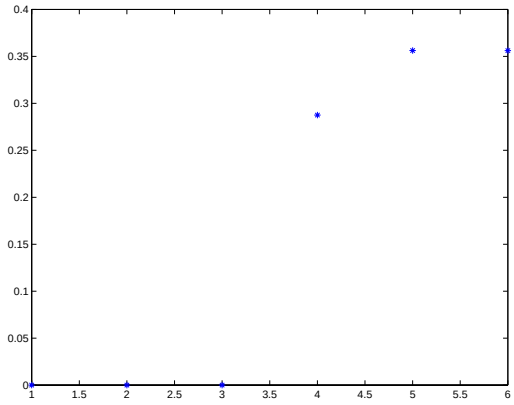
Iteration 2



Iteration 3



Iteration 4



Iteration 5

Figure 4.2: Evolution of the state vector $\mathbf{x}(t)$ for the points in Figure 4.1, using the first-order discrete-time replicator model, equation (4.11).

Chapter 5

Results on Image Segmentation and Shape Categorization

In this chapter we apply our partitional clustering algorithm to some fundamental computer vision problems, namely image segmentation and shape categorization. In Section 5.1 we consider segmentation of intensity, color and textured images. In Section 5.2 we apply our clustering methodology to the problem of the unsupervised learning of shape categories from an image database. Specifically, two different collections of images reported in the literature are divided into shape categories.

5.1 Image Segmentation

For image segmentation, nodes in the graph represent individual pixels. The difference between the segmentation of brightness, color and textured images lies solely in the similarity measure between pixels. While brightness is a one dimensional attribute, the color resemblance between pixels is measured in a three dimensional color space and texture resemblance is measured on a higher dimensional feature space. In accordance, the similarity weight $w(i,j)$ between pixels (nodes) i and j increases with increasing brightness/color/texture likelihood. Specifically, in our experiments the similarity between pixels i and j was measured by (see equation 2.6):

$$w(i,j) = \exp\left(\frac{-\|\mathbf{F}(i) - \mathbf{F}(j)\|_2^2}{\sigma^2}\right)$$

where σ is a positive real number which affects the decreasing rate of w, and $\mathbf{F}(i)$ is a feature vector based on intensity, color or texture information at node i, defined in Sections

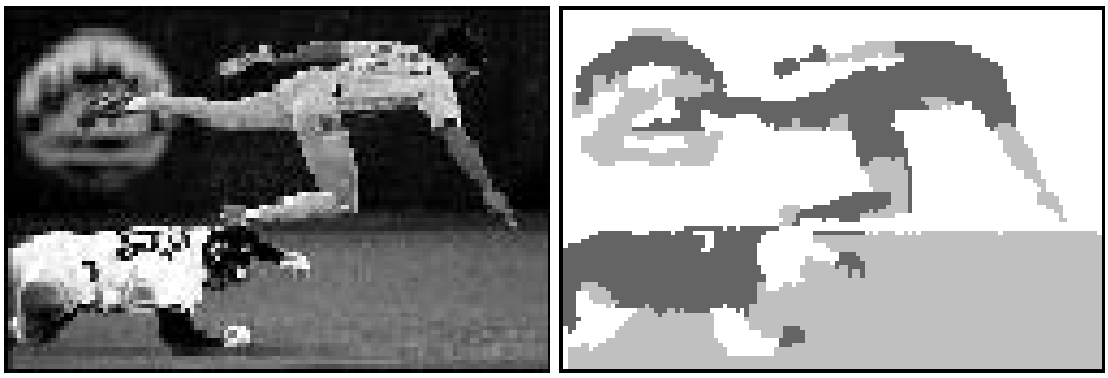


Figure 5.1: An 83×125 intensity image (left) and its segmentation (right). Parameter setting: $\sigma = 0.14$. Minimal region size of interest is 11.

5.1.1, 5.1.2 and 5.1.3 respectively. As customary, we determine the parameter σ manually. We refer the reader to [102] for an empirical comparison of dissimilarity measures for color and texture.

In principle, our clustering algorithm consists of iteratively finding a dominant set in the graph using replicator dynamics and then removing it from the graph, until all vertices have been clustered. Indeed, in the experiments reported here, to avoid the formation of small meaningless clusters, we repeated the process until 90% of the pixels were clustered, the remaining ones being assigned to the closest clusters, in terms of brightness/color/texture proximity. In our experiments, we used the discrete-time first-order replicator equations (4.11). The process was started from the simplex barycenter and stopped after a few iterations (typically, no more than three). To improve the segmentation results, after convergence small isolated regions were incorporated into larger ones (see figure's captions for details).

5.1.1 Intensity Image Segmentation

We consider images where each pixel has a single intensity attribute. In accordance, $\mathbf{F}(i)$ is defined as the intensity value at node i, normalized to a real number in the interval $[0, 1]$.

Figure 5.1 shows an intensity image taken during a baseball game that has been used originally by Shi and Malik [112] and other authors [23, 32]. Here and in the sequel, the left

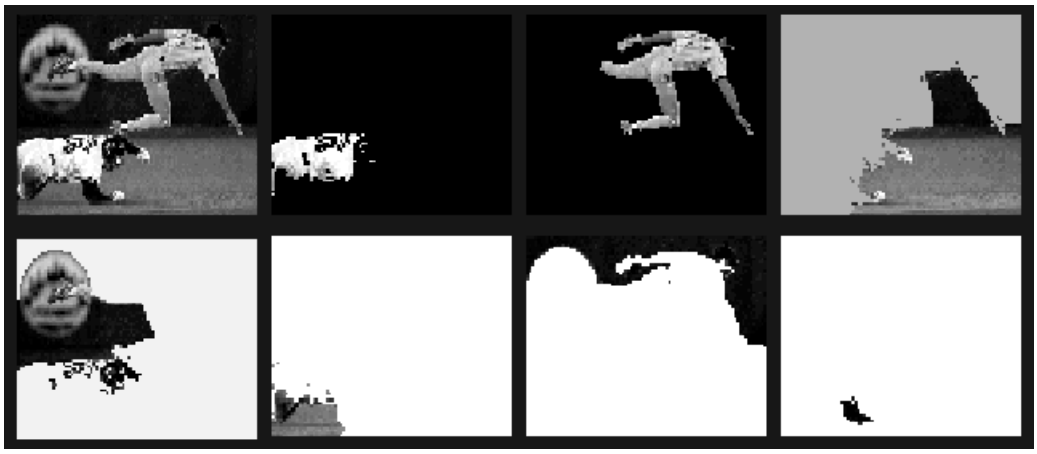


Figure 5.2: Normalized cut algorithm result [112].

side of the figures reporting our results shows the original image and the right one shows the corresponding segmentation, where connected pixels having the same gray level are intended to belong to the same region. As can be seen, unlike other algorithms [112, 32, 23] (see Figures 5.2 to 5.4) ours was able to separate the grassy region from the back wall in a nice way. The uniforms of the two players (which have significant variation due to folds in the cloth) are also segmented in a satisfactory way, and smaller but important components such as the arms and gloves are also correctly segmented. Like other algorithms [112, 32], however, ours did not succeed in distinguishing between the Mets emblem and the left leg of the top player. Also, note that the helmet of the bottom player is incorrectly merged with the back wall. Overall, these results compare well with those presented in [112, 32], and are substantially better than those obtained with an optimally-tuned standard split-and-merge algorithm [32] (see Figure 5.5).

Figure 5.6, which shows a weather radar image, has also been used in [112] as an instance whereby edge-detection-based segmentation would perform poorly. The algorithm was able to correctly partition the image into a background and a foreground, and our results compare well with those reported in [112], where the image was separated into more components (see Figure 5.7).

Figure 5.8 shows the facade of a building. Again, the segmentation produced by our algorithm is of good quality. Specifically, the sky, the windows, the banners, and the wall (except from some noisy components around the central and right window on the first level) are all correctly segmented. Note also that the shadow of the balcony, the central

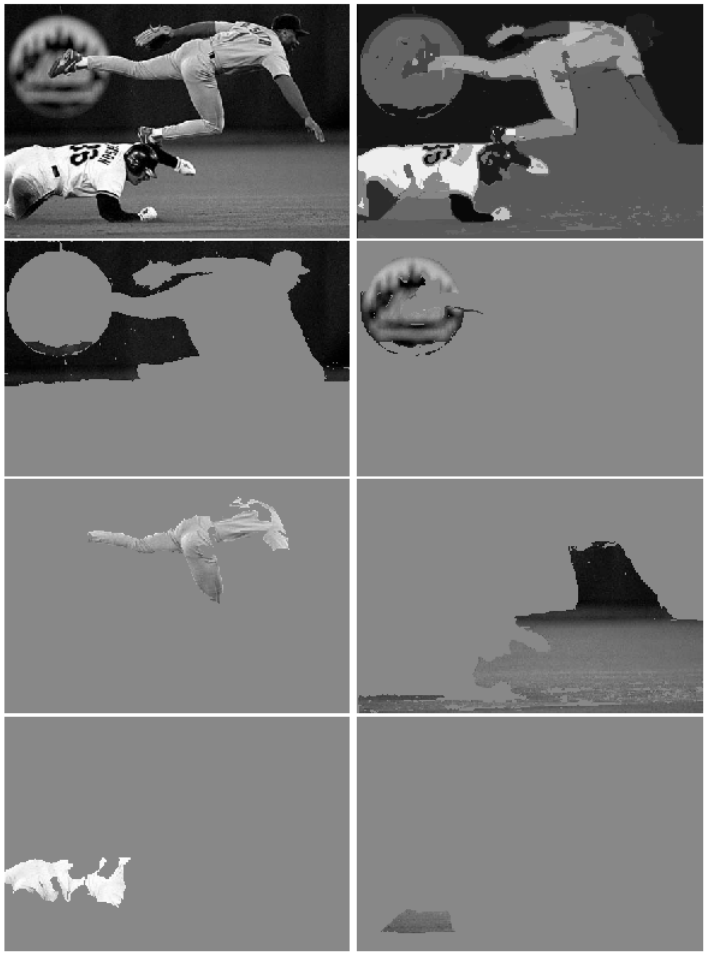


Figure 5.3: Felzenszwalb and Huttenlocher [23].

Figure 5.4: Typical cut algorithm results (from [32]).

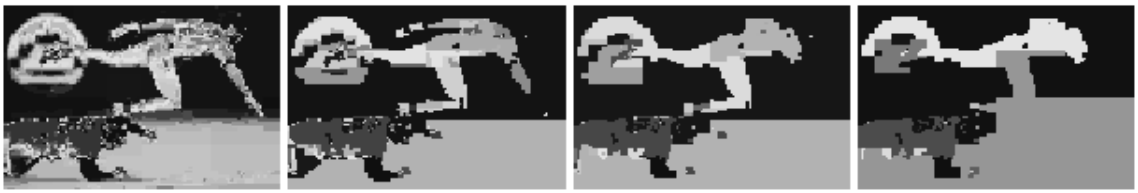


Figure 5.5: Results obtained with an optimally-tuned split-and-merge algorithm (from [32]).

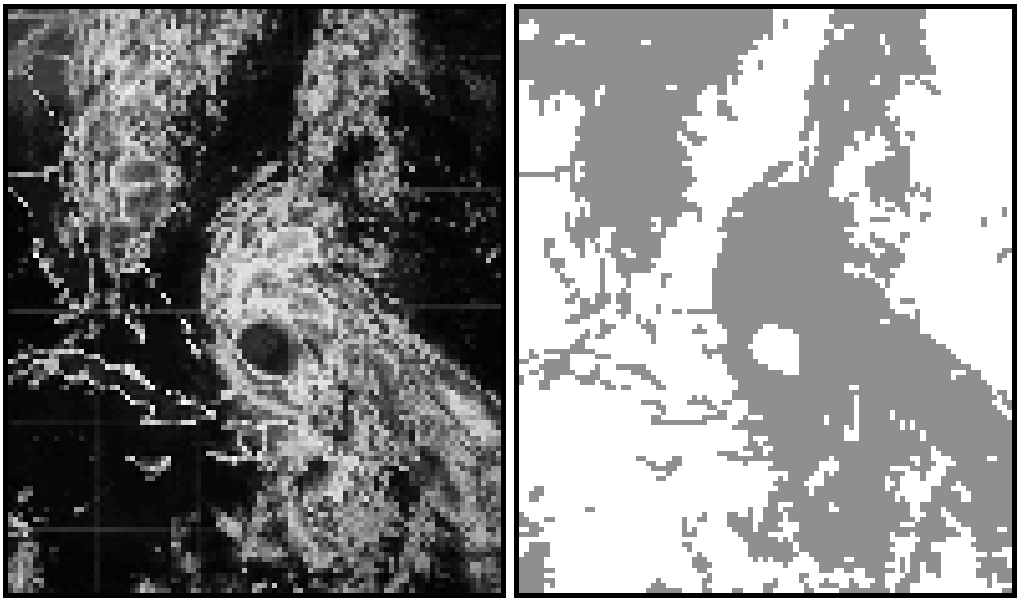


Figure 5.6: A 115×97 intensity image (left) and its segmentation (right). Parameter settings: $\sigma = 0.1$. Minimal region size of interest is 2.

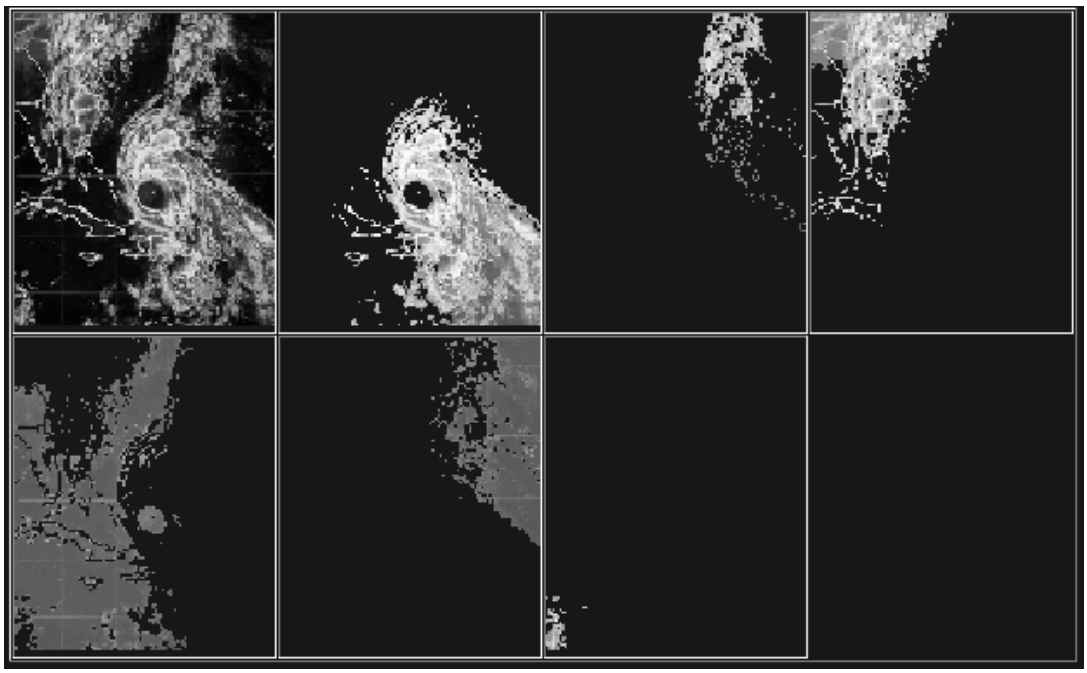


Figure 5.7: Normalized cut algorithm result (from [112]).

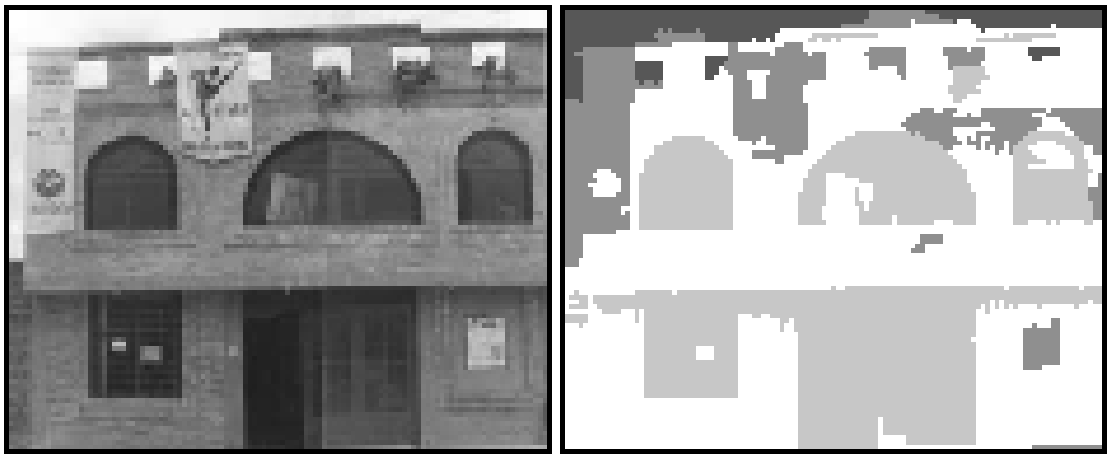


Figure 5.8: A 94×115 intensity image (left) and its segmentation (right). Parameter setting: $\sigma = 0.15$. Minimal region size of interest is 11.

door, and the window on the left-hand side are erroneously put in a unique component.

5.1.2 Color Image Segmentation

Here we consider images where each pixel has three color attributes. Specifically, $\mathbf{F}(i)$ is defined as the $\mathbf{F}(i) = [v, vs\sin(h), vs\cos(h)](i)$, where h, s, v are the HSV values [26] of pixel i.

Figures 5.9 and 5.10 show results on color images taken from the COREL database. Figure 5.9 shows the image of a spaceman. Note how the segmentation is very clean despite substantial color variation. Essentially, the algorithm found three large regions: a large component for the black sky, another one for the earth, and a third one for the astronaut. Note also that some spurious regions were also found in the earth and in the man areas.

Figure 5.10 (used also in [23]) shows an image of the Eiffel tower at night. Here, the algorithm was able to partition the image into meaningful components. It found a large component for the sky and, within the tower, it distinguished between the bright and the dark areas. However, note that the dark area in the bottom part of the image was incorrectly merged with the lower dark part of the tower. The quality of our segmentation looks comparable to the one reported by Felzenszwalb and Huttenlocher in [23] (see

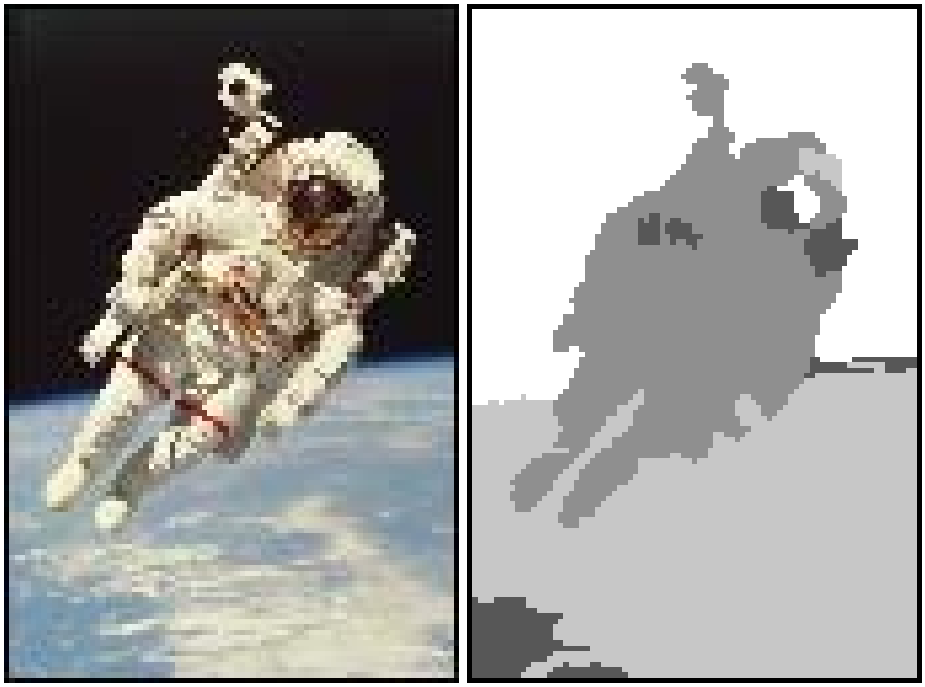


Figure 5.9: An 125×83 color image (left) and its segmentation (right). Parameter setting: $\sigma = 0.11$. Minimal region size of interest is 16.

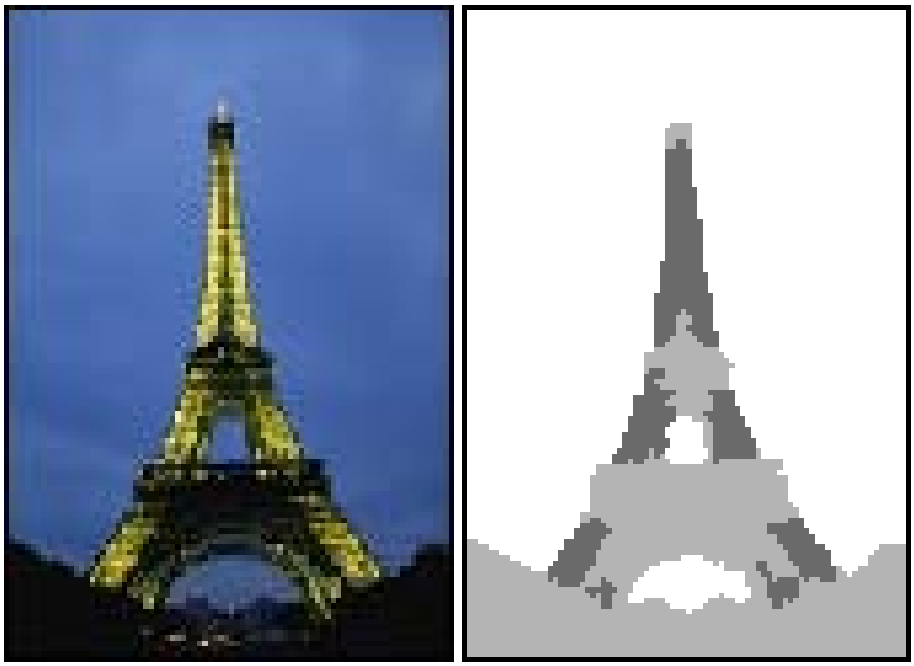


Figure 5.10: A 121 × 82 color image (left) and its segmentation (right). Parameter settings: $\sigma = 0.325$. Minimal region size of interest is 10.

Figure 5.11): they were able to separate the lower dark area from the tower, but failed in clearly distinguishing the dark from the bright tower regions.

5.1.3 Textured Image Segmentation

Here $\mathbf{F}(i)$ is a feature vector based on texture information at pixel i defined as $\mathbf{F}(i) = [|I * f_1|, \ldots, |I * f_k|](i)$ and the f_i are DOOG filters at various scales and orientations as used in [70]; the intensity values I_i are intended to be normalized to real numbers in the interval [0,1].

Figures 5.12 to 5.14 show the results obtained with our segmentation algorithm on various textured images. Figure 5.12 shows a stripe of textures taken from the USC-SIPI image database (avaiable at www.sipi.usc.edu/services/database). In this example, to improve the segmentation result, a Gaussian smoothing on the filter responses was performed. The process was stopped after 5 iterations.

Figures 5.13 and 5.14 show segmentation results for two natural images of a zebra and a cheetah against a noisy background. In both cases, despite of significant variation of the

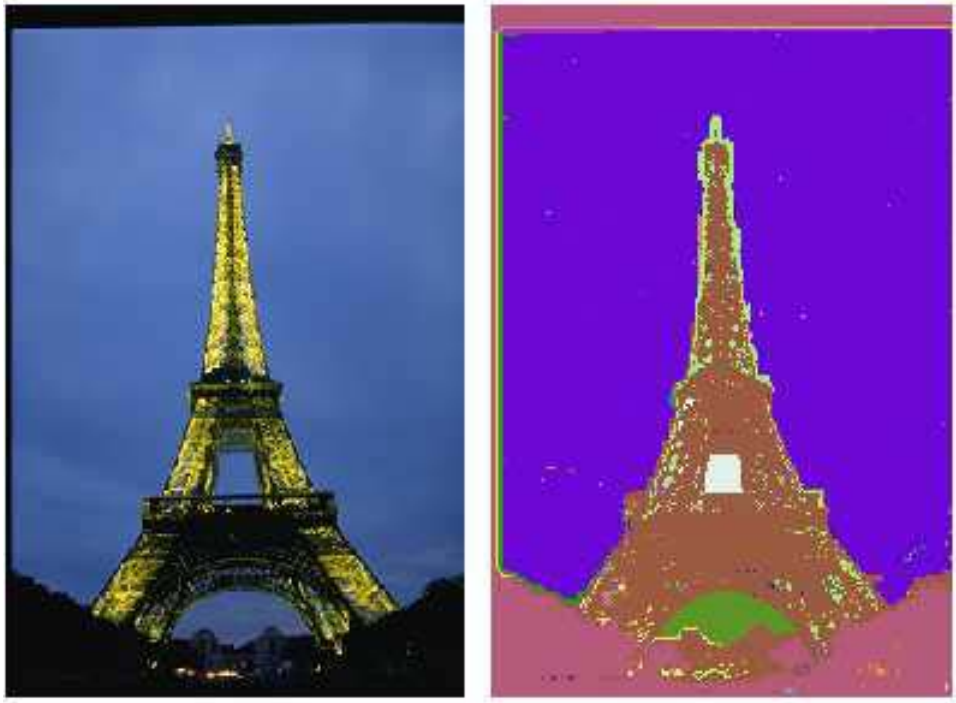


Figure 5.11: Felzenszwalb and Huttenlocher (from [23]).

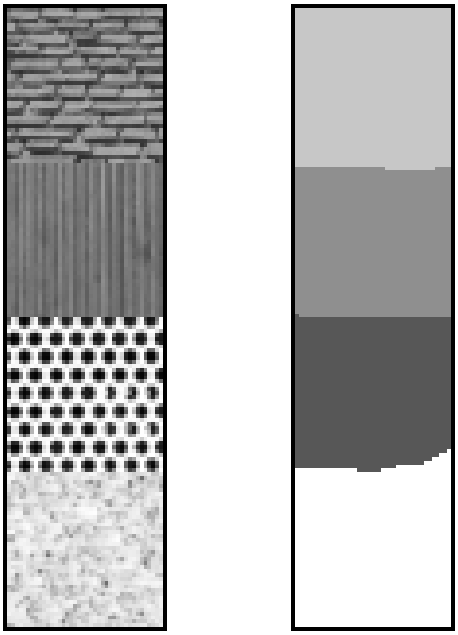


Figure 5.12: A stripe of four 40×40 texture patches (left) and its segmentation (right). Parameter setting: $\sigma = 0.02$.

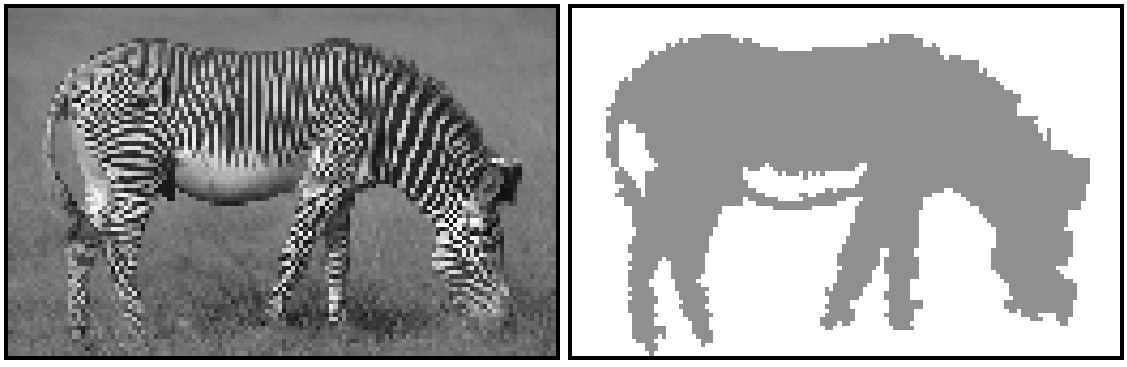


Figure 5.13: An 81×128 zebra image (left) and its segmentation (right). Parameter setting: $\sigma = 0.1$.

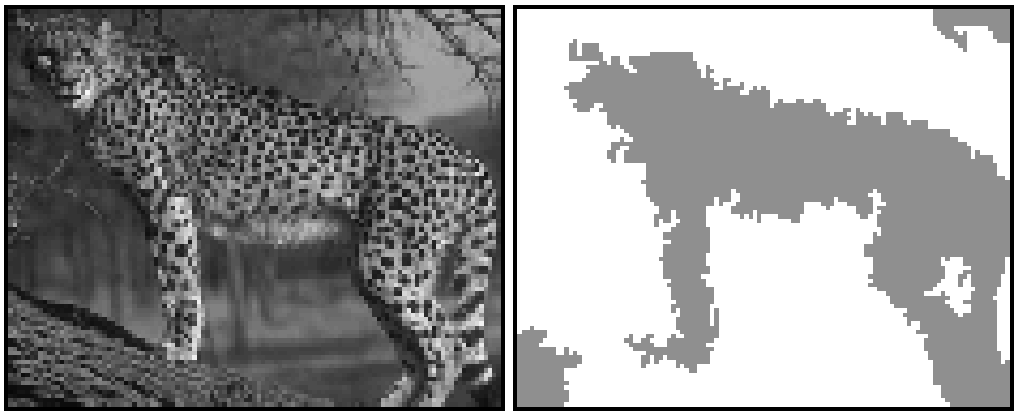


Figure 5.14: A 93×115 cheetah image (left) and its segmentation (right). Parameter setting: $\sigma = 0.17$.

textures on the body of the animals, our algorithm was able to capture the big picture of the scenes. On average, the segmentation algorithm took only a few seconds on an Intel Pentium III 750 Mhz.

5.2 Shape Categorization

We now illustrate the potential of our approach on the problem of the unsupervised learning of shape categories from an image database. To this end, we use two different similarity matrices reported in the literature derived from as many binary shape databases.

5.2.1 Using Luo et al.'s Similarities

The first dataset we applied our algorithm on was used by Luo et al. [67], who developed a maximum-likelihood framework for graph clustering. The dataset contains 25 different shapes from 9 different classes: brushes, wrenches, pliers, hammers, spectacles, fishes, rabbits, horses, and hands.

In the work described in [67], shapes were abstracted in terms of rooted shock-trees [113], and the similarities between pairs of shapes were computed by using the weighted tree edit distance of the corresponding trees. The resulting similarity matrix (whose entries are taken from [67, Fig. 1] and scaled properly) is shown in Figure 5.15 (left). The order of the entries in the matrix is the same as the one given above.

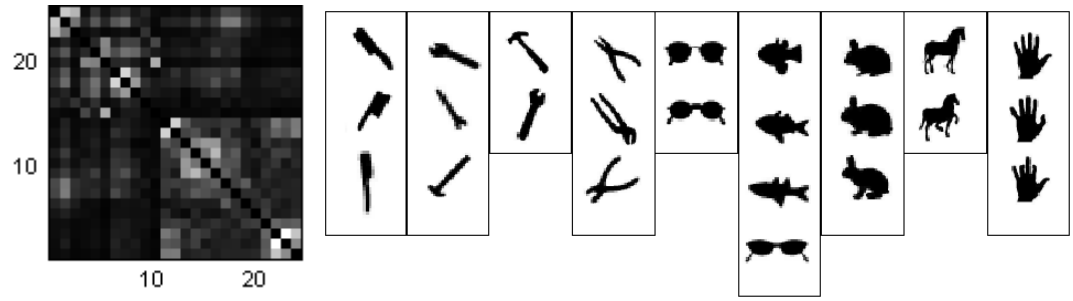


Figure 5.15: Left: Similarity matrix used in the experiments described in Sect. 5.2.1. Right: The nine categories discovered by our algorithm.

Figure 5.16: Luo et al. (from [67]).

The results obtained using our clustering algorithm on this dataset are shown in Fig. 5.16 (right). The algorithm discovered (correctly) 9 classes, and made only 3 mistakes by placing a pair of glasses in the fish class, and by swapping two elements from the hammer and wrench classes. These results are substantially better than those reported by Luo et al. in [67] (see Figure 5.16). Indeed, they were able to partition the database into 7 (instead of 9) classes, and made more classification errors, by placing two fishes in the spectacle class, a hammer in the wrench classes, and by putting together in a quite inhomogeneous cluster, a hammer, a fish, a rabbit, and the two horses.

5.2.2 Using Klein et al.'s Similarities

The second series of experiments was conducted on the database used by Klein et al. in [59], which contains 36 binary shapes from 6 different categories: fishes, wrenches, planes, "greebles," rabbits, and hands. Each category contains 6 different shapes.

They used an unrooted and ordered shock-tree representation for shapes. To measure the distance between two shapes, they first compared each path in one shock graph to each path in the other, computing a cost of deforming one to the other, and then used these costs to compute the edit-distance. Fig. 5.17 (left) shows the similarity matrix obtained by transforming the distances presented in [59, Table 1] into affinities using a customary exponential transformation (see equation 2.6). The order of the entries is the same as the one given above.

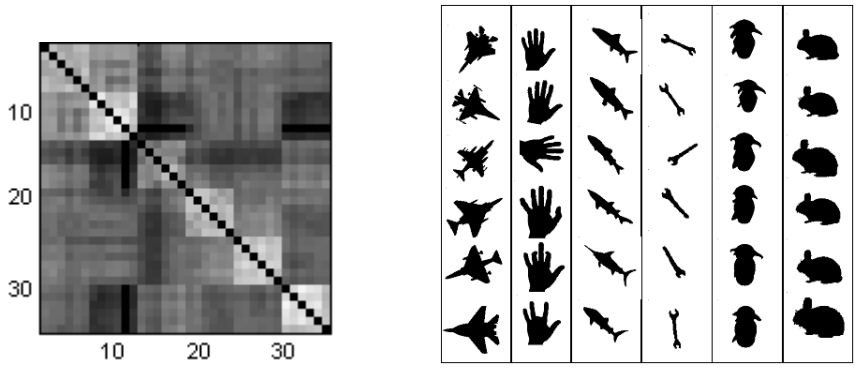


Figure 5.17: Left: Similarity matrix used in the experiments described in Sect. 5.2.2. Right: The six categories discovered by our algorithm.

Fig. 5.17 (right) shows the results obtained using our algorithm, which partitioned the dataset into the right number of classes and made no classification error.

Chapter 6

Dominant Sets and Hierarchical Clustering

In Chapters 2 to 5 we have developed a new framework for partitional pairwise clustering based on a new graph-theoretic concept, that of a *dominant set*, which generalizes the notion of a maximal clique to edge-weighted graphs. A connection between dominant sets and the solutions of a (continuous) quadratic optimization problem allows us the use of straightforward dynamics from evolutionary game theory to determine them. We have shown in Chapter 5 how the approach is a powerful one when applied to problems such as image segmentation and shape categorization.

However, in many computer vision applications, such as the organization of an image database [106, 7], it is important to organize the data to be clustered in a hierarchical manner. In this chapter we show how to do this within the dominant set framework. To this end, we consider a family of regularized (continuous) quadratic programs controlled by a nonnegative parameter which determines the global shape of the energy landscape as well as the location of its extrema. We investigate the properties of its solutions as a function of its parameter. When the regularization parameter is zero the local solutions are known to be in one-to-one correspondence with dominant sets, but when it is positive an interesting picture emerges. As the parameter grows larger, local solutions corresponding to small clusters disappear, and we derive bounds for it that allow us to exclude from the set of solutions those inducing clusters of size smaller than a prescribed threshold. This suggests a new (divisive) hierarchical approach to clustering, which is based on the idea of properly varying the regularization parameter during the clustering process. We start with a sufficiently large value, which yields a unique large cluster comprising all data, and then decrease it properly in an attempt to split large and incoherent clusters into smaller

pieces. The process is repeated recursively at each level in the hierarchy, where replicator dynamics decribed in Chapter 4 are used as local optimization procedures.

We start in Section 6.1 by introducing a family of quadratic programs which contain a parameterized regularization term. Bounds for the regularization parameter are derived in Section 6.2. The description of the algorithm follows in Section 6.3.

6.1 A Family of Quadratic Programs

Let $A = (a_{ij})$ be the similarity matrix of the n data to be clustered, and consider the following family of standard quadratic programs:

$$\begin{aligned} \text{maximize} \quad & f_\alpha(\mathbf{x}) = \mathbf{x}^{\mathrm{T}}(A - \alpha I)\mathbf{x} \\ \text{subject to} \quad & \mathbf{x} \in \Delta \end{aligned} \tag{6.1}$$

where $\alpha \geq 0$ is a parameter and I is the identity matrix, which includes as special case program (3.1) when $\alpha = 0$.

Note that the solutions of (6.1) remain the same if the matrix $A - \alpha I$ is replaced with $A - \alpha I + \kappa \mathbf{e}\mathbf{e}^{\mathrm{T}}$, where κ is an arbitrary parameter, since $\mathbf{x}^{\mathrm{T}}(A - \alpha I + \kappa \mathbf{e}\mathbf{e}^{\mathrm{T}})\mathbf{x} = \mathbf{x}^{\mathrm{T}}(A - \alpha I)\mathbf{x} + \kappa$ for all $\mathbf{x} \in \Delta$. In particular, if $\kappa = \alpha$ the resulting matrix is nonnegative and has a null diagonal. Hence, Theorem 3.1 applies and all (strict) solutions of (6.1) correspond to dominant sets for the scaled similarity matrix $A + \alpha(\mathbf{e}\mathbf{e}^{\mathrm{T}} - I)$ having the off-diagonal entries equal to $a_{ij} + \alpha$.

A point $\mathbf{x} \in \Delta$ satisfies the KKT conditions for problem (6.1) if there exist $n + 1$ real constants $\mu_1, \ldots \mu_n$ and λ, with $\mu_i \geq 0$ for all $i = 1 \ldots n$, such that:

$$(A\mathbf{x})_i - \alpha x_i - \lambda + \mu_i = 0$$

for all $i = 1 \ldots n$, and

$$\sum_{i=1}^{n} x_i \mu_i = 0 \ .$$

Note that, since both the x_i's and μ_i's are nonnegative, the latter condition is equivalent to saying that $i \in \sigma(\mathbf{x})$ implies $\mu_i = 0$. Hence, the KKT conditions can be rewritten as

$$(A\mathbf{x})_i - \alpha x_i \begin{cases} = \lambda & \text{if } i \in \sigma(\mathbf{x}) \\ \leq \lambda & \text{otherwise} \end{cases}$$

for some real constant λ. On the other hand, it is immediate to see that $\lambda = \mathbf{x}^{\mathrm{T}}(A-\alpha I)\mathbf{x}$. Hence the previous conditions can be explicitly rewritten as

$$\begin{array}{lcl} (A\mathbf{x})_i - \alpha x_i & = & \mathbf{x}^{\mathrm{T}} A\mathbf{x} - \alpha \mathbf{x}^{\mathrm{T}}\mathbf{x}, \text{ if } i \in \sigma(\mathbf{x}) \\ (A\mathbf{x})_i & \leq & \mathbf{x}^{\mathrm{T}} A\mathbf{x} - \alpha \mathbf{x}^{\mathrm{T}}\mathbf{x}, \text{ otherwise} \end{array} \tag{6.2}$$

As thoroughly discussed in Chapter 4, a straightforward way to find (local) solutions of program (6.1) is given by the so-called (discrete-time) first-order replicator dynamics:

$$x_i(t+1) = x_i(t)\frac{(A\mathbf{x})_i - \alpha x_i(t)}{\mathbf{x}(t)^{\mathrm{T}}(A-\alpha I)\mathbf{x}(t)} \tag{6.3}$$

for $i = 1 \ldots n$.

Summarizing the results reported in Chapter 4, we have that, provided that the matrix $A - \alpha I$ is scaled properly to avoid negative values, the simplex Δ is invariant under these dynamics. Moreover, the (scaled) objective function f_α is strictly increasing along any nonconstant trajectory of (6.3), and its asymptotically stable points are in one-to-one correspondence to strict local solutions of (6.1). These, in turn, correspond to dominant sets for the scaled similarity matrix $A + \alpha(\mathbf{e}\mathbf{e}^{\mathrm{T}} - I)$.

6.2 Bounds for the Regularization Parameter

The objective function f_α in (6.1) consists of a data term and a regularization term. The first one ($\mathbf{x}^{\mathrm{T}} A\mathbf{x}$) favors solutions with high internal coherency, and the second ($-\alpha\mathbf{x}^{\mathrm{T}}\mathbf{x}$), which is controlled by the regularization parameter α, acts as an entropic factor: it is concave and, on the simplex Δ, it is maximized at the barycenter and it attains its minimum value at the vertices of Δ. Hence, when α is large enough the regularization term dominates, and the only solution of (6.1) is expected to be in the interior of Δ, probably close to the barycenter. In other words, we expect a unique large cluster which comprises all the data points.

This intuitive picture is formalized by the following proposition, which gives also a simple bound for the regularization parameter α. First, however, note that since A is symmetric, all its eigenvalues are real, and we shall denote by $\lambda_{\max}(A)$ the largest eigenvalue of A.

Proposition 6.1 *If $\alpha > \lambda_{\max}(A)$, then f_α is a strictly concave function in $\mathbb{R}^n$, and the only solution $\mathbf{x}$ of (6.1) belongs to $\mathrm{int}(\Delta)$, i.e., $\sigma(\mathbf{x}) = V$.*

Proof. We have $\lambda_{\max}(A - \alpha I) = \lambda_{\max}(A) - \alpha < 0$. Hence the matrix $A - \alpha I$ is negative definite, which implies that f_α is strictly concave. Consequently, program (6.1) has a unique solution, say $\mathbf{x}$. Suppose by contradiction that $\mathbf{x}$ lies on the boundary of Δ, i.e., $\sigma(\mathbf{x}) \neq V$. From the KKT conditions (6.2) we have:

$$(A\mathbf{x})_i \leq \mathbf{x}^{\mathrm{T}} A\mathbf{x} - \alpha \mathbf{x}^{\mathrm{T}}\mathbf{x} < 0$$

for all $i \in V \setminus \sigma(\mathbf{x})$, which is absurd since A and $\mathbf{x}$ are nonnegative. □

For smaller values of α, however, it is difficult to predict what happens to the landscape of f_α, as it is not obvious how the data and the regularization terms interact. The next theorem provides an answer to this question.

Theorem 6.1 *Let $S \subset V$ be a proper subset of vertices ($S \neq V$), and let A_S denote the submatrix of A formed by the rows and columns indexed by the elements of S. If*

$$\alpha > \lambda_{\max}(A_S)$$

then there is no point $\mathbf{x} \in \mathrm{int}(\Delta_S)$ that is a local maximizer of f_α in Δ.

Proof. Let $\alpha > \lambda_{\max}(A_S)$. We first show that for all $\mathbf{x} \in \mathrm{int}(\Delta_S)$ we have:

$$-\gamma(\mathbf{x}) \leq \lambda_{\max}(A_S)$$

where

$$\gamma(\mathbf{x}) = \max_{i \notin \sigma(\mathbf{x})} \frac{(A\mathbf{x})_i - \mathbf{x}^{\mathrm{T}} A\mathbf{x}}{\mathbf{x}^{\mathrm{T}}\mathbf{x}} \ .$$

Indeed, for $\mathbf{x} \in \mathrm{int}(\Delta_S)$, let $\mathbf{x}_S$ be the vector obtained from $\mathbf{x}$ by dropping all components in $V \setminus S$. We have:

$$\begin{aligned} \gamma(\mathbf{x}) &= \max_{i \notin \sigma(\mathbf{x})} \frac{(A_S\mathbf{x}_S)_i}{{\mathbf{x}_S}^{\mathrm{T}}\mathbf{x}_S} - \frac{{\mathbf{x}_S}^{\mathrm{T}} A_S {\mathbf{x}_S}^{\mathrm{T}}}{{\mathbf{x}_S}^{\mathrm{T}}\mathbf{x}_S} \\ &\geq \max_{i \notin \sigma(\mathbf{x})} \frac{(A_S\mathbf{x}_S)_i}{{\mathbf{x}_S}^{\mathrm{T}}\mathbf{x}_S} - \lambda_{\max}(A_S) \\ &\geq -\lambda_{\max}(A_S) \end{aligned}$$

where the first inequality follows from the Rayleigh-Ritz theorem [51], and the last one from the nonnegativity of A and $\mathbf{x}$.

Now, suppose by contradiction that there exists a point $\mathbf{x}$ in $\mathrm{int}(\Delta_S)$ which is a local maximizer of $f_\alpha(\mathbf{x})$ in Δ. Since $-\alpha < -\lambda_{\max}(A_S) \leq \gamma(\mathbf{x})$ there exists an $i \notin \sigma(\mathbf{x}) = S$ such that

$$-\alpha < \frac{(A\mathbf{x})_i - \mathbf{x}^{\mathrm{T}} A\mathbf{x}}{\mathbf{x}^{\mathrm{T}}\mathbf{x}}$$

which yields $\mathbf{x}^{\mathrm{T}} A\mathbf{x} - \alpha \mathbf{x}^{\mathrm{T}}\mathbf{x} < (A\mathbf{x})_i = (A\mathbf{x})_i - \alpha x_i$. This means that the KKT conditions (6.2) are violated, and this contradicts the hypothesis that $\mathbf{x}$ is a solution of (6.1). □

Therefore, suppose that S is a cluster (i.e., a dominant set) we want to avoid. By letting $\alpha > \lambda_{\max}(A_S)$ no point in Δ with support S will be a solution of (6.1). Of course, the problem is to obtain a reasonable bound for α without knowing S in advance. To this end, suppose for simplicity that $a_{ij} \leq 1$ for all $i, j \in V$, namely

$$0 \leq A \leq \mathbf{e}\mathbf{e}^T - I \ .$$

From standard results on nonnegative matrices (see, e.g., [51]), we get

$$\lambda_{\max}(A_S) \leq \lambda_{\max}(\mathbf{e}\mathbf{e}^T - I) = |S| - 1$$

for any $S \subseteq V$. Hence, if we want to avoid clusters of size $|S| \leq m < |V|$ we could simply let

$$\alpha > m - 1 \ .$$

In so doing, no face Δ_S with $|S| \leq m$ will contain solutions of (6.1). Hence, these faces cannot be approached by any interior trajectory of replicator dynamics with payoff $A - \alpha I$; in other words, at this scale, all clusters will have at least $m + 1$ data points.

On the other hand, by virtue of Proposition 6.1, if

$$\alpha > |V| - 1 \geq \lambda_{\max}(A)$$

we always get a unique large cluster comprising all points in V.

6.3 The Hierarchical Clustering Algorithm

Summarizing the above findings, when $\alpha > m - 1$ the energy landscape of f_α is populated only by solutions having a support with more than m data points. Among them, some will correspond to dominant sets for the original matrix A and some will be "spurious" solutions, namely, solutions of program (6.1) that are not characteristic vectors of a dominant set for A. Spurious solutions represent large subsets of points that are not sufficiently coherent to be dominant with respect to A, although they are dominant for the scaled matrix $A + \alpha(\mathbf{e}\mathbf{e}^{\mathrm{T}} - I)$, and hence they should be split. Indeed, it is precisely the emergence of these spurious solutions that allows us to provide the data to be clustered with a natural hierarchical organization.

Instead of keeping the value of α fixed, our approach is to start with a sufficiently large α, say $\alpha > \lambda_{\max}(A)$, and adaptively decrease it during the clustering process. The rationale behind this idea is that for values of α that are large enough, only the characteristic vectors of large dominant sets will be stable attractive points for the replicator dynamics, together with a set of spurious solutions. As the value of α decreases, spurious solutions disappear and at the same time (characteristic vectors of) smaller dominant sets become stable. At a given level α, the algorithm produces a (flat) partition of the data set into several "clusters," some of which will be dominant with respect to the unscaled similarities (and we declare them leaves of our hierarchy) and some will not. We then proceed by decreasing α and recursively applying the procedure to these spurious solutions, in an attempt to split them into smaller and more coherent pieces.

A high-level description of our algorithm is shown in Fig. 6.1 (as usual, V is the set of data points to be clustered and A is the corresponding similarity matrix). Its output is in the form of nested sets, with the nesting providing the hierarchy.

A few remarks about the previous algorithm are in order. The function SPLIT accepts as input a set of vertices, the corresponding similarity matrix, and a parameter α, and provides as output a (flat) partition of the input set at level (or scale) α. This can be accomplished by iteratively finding a local maximizer of f_α in Δ (using, for example, replicator dynamics with payoff $A - \alpha I$) and then removing the vertices in its support from the input set, until all vertices have been clustered.

Note also that the repeat-until loop must terminate. In fact, since V is not dominant

Algorithm HIER_CLUSTERING(V, A)
begin
  if V is dominant (or is a singleton) then return V
  let α be a large positive value (e.g., $\alpha > |V| - 1$)
  repeat
    decrease α (e.g., $\alpha \leftarrow \alpha - 1$)
    if $\alpha < 0$ then $\alpha \leftarrow 0$
    $V_1, \ldots, V_k \leftarrow$ SPLIT(V, A, α)
  until $k > 1$
  return $\bigcup_{i=1}^{k}$ { HIER_CLUSTERING(V_i, A_{V_i}) }
end

Figure 6.1: Pseudo-code for our hierarchical clustering algorithm.

(and is not a singleton) the function SPLIT will certainly return a non-trivial partition of V for all α's in some interval $[0, \hat{\alpha}]$.

Finally, note that checking that the set V is dominant is straightforward. In fact, this happens if and only if there exists a (unique, see [11]) strict local solution of (3.1) in $\text{int}(\Delta)$ and this happens if and only if any interior trajectory or replicator dynamics with payoff A converges, with probability one, to a point in $\text{int}(\Delta)$.

Chapter 7

Results on Image Database Organization

In this chapter we apply our hierarchical clustering algorithm to the problem of organizing an image database. This problems deal with the hierarchical aggregation of visual elements, namely contours extracted from images, into coherent groups.

7.1 Organizing a Shape Database

According to the general approach adopted in this work, our first step is to represent the images as nodes in a graph, and assign a similarity weight $w(i, j)$ to each edge. To this end, we used three different similarity matrices reported in the literature derived from as many binary shape databases.

7.1.1 Using Luo et al.'s Similarities

The first dataset we applied our hierarchical clustering algorithm on was used by Luo et al. [67] (already described in some detail in Section 5.2.1). The dataset contains 25 different shapes from 9 different classes: brushes, wrenches, pliers, hammers, spectacles, fishes, rabbits, horses, and hands. The similarity matrix is shown again in Fig. 7.1 (left). The order of the entries in the matrix is the same as the one given above.

The results obtained using our hierarchical clustering algorithm on this dataset are shown in Fig. 7.1 (right). At the lowest level of the hierarchy, the algorithm discovered (correctly) 9 classes, and made only 3 mistakes by placing a pair of glasses in the fish class, and by swapping two elements from the hammer and wrench classes (these results are identical to those produced by the partitional algorithm, Section 5.2.1). As for the

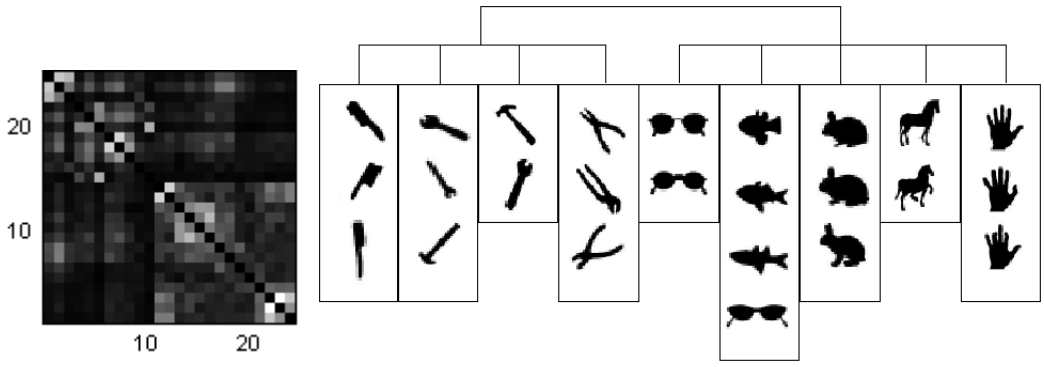


Figure 7.1: Left: Similarity matrix used in the experiments described in Section 7.1.1. Right: Hierarchy produced by our algorithm.

upper levels of the hierarchy, the algorithm was able to separate the tools from the rest of the objects. Note that these data do not exhibit a natural hierarchical organization, hence it is not surprising that the algorithm produced a shallow tree.

7.1.2 Using Klein et al.'s Similarities

A second series of experiments was conducted on the database used by Klein et al. in [59] (described in some detail in Section 5.2.2), which contains 36 binary shapes from 6 different categories: fishes, wrenches, planes, "greebles," rabbits, and hands. Each category contains 6 different shapes. Fig. 7.2 (left) shows the similarity matrix. The order of the entries is the same as the one given above.

Fig. 7.2 (right) shows the results obtained using our algorithm. Starting from the root of the hierarchy, the algorithm first distinguished between planes and hands from the rest of the dataset, and it further refined the latter classes by separating fishes and tools from greebles and rabbits. At the base level, the algorithm partitioned the dataset into the right number of classes and made no classification error (as the partitional algorithm did, Section 5.2.2). Note that, although some of these high-level groupings (e.g., tools and fishes) make little sense from a semantic standpoint, the corresponding shapes are indeed topologically similar, as can also be seen by manually inspecting the similarity matrix.

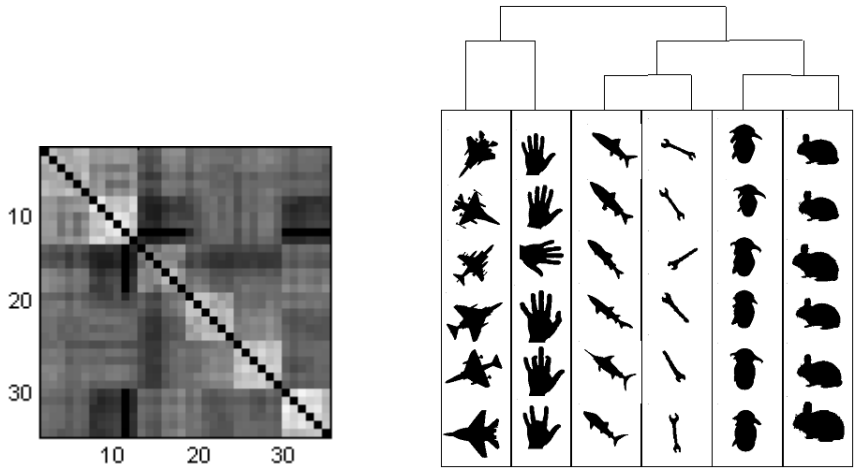


Figure 7.2: Left: Similarity matrix used in the experiments described in Section 7.1.2. Right: Hierarchy produced by our algorithm.

7.1.3 Using Gdalyahu et al.'s Similarities

Finally, we applied our algorithm on a larger database used by Gdalyahu et al. in a series of papers [30, 31, 32] which contains 121 binary images, 90 of which represent 6 toy models seen from 15 different viewpoints, while the other ones are 31 silhouettes taken from the database used in the previous section. Overall, there are 12 categories: cows, hippos, wolves, cars, sport cars, children, hands, fishes, planes, rabbits, wrenches, and greebles.

In [31], the (dis)similarities between shapes were computed by constructing a syntactic representation for the shape boundaries and then finding an edit transformation which maps one curve to the other by dynamic programming. Similarities were then obtained by a standard exponential transformation (see equation 2.6). The resulting matrix is shown in Fig. 7.3 (top), with the entries ordered as above.

The results of applying our algorithm on this database are shown in Fig. 7.3 (bottom). Here, for each base-level cluster we show a representative for each category contained in it and, in parenthesis, the number of objects in that category. At the coarsest scale, the algorithm first distinguished the cars from all other classes and, by proceeding down along the car branch, it then correctly separated the two car models, although it created an extra spurious cluster with 5 cars. As for the remaining objects, it separated the mammals (comprising the children) from the other objects, erroneously putting the wrenches in the

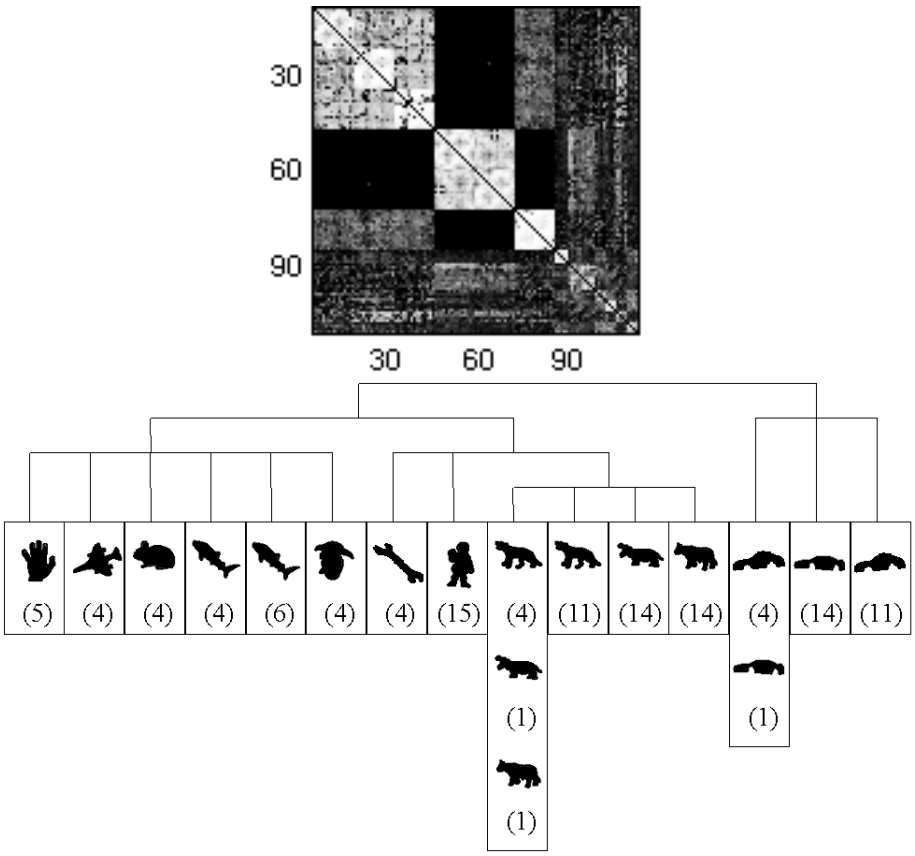


Figure 7.3: Top: Similarity matrix used in the experiments described in Section 7.1.3. Bottom: Hierarchy produced by our algorithm.

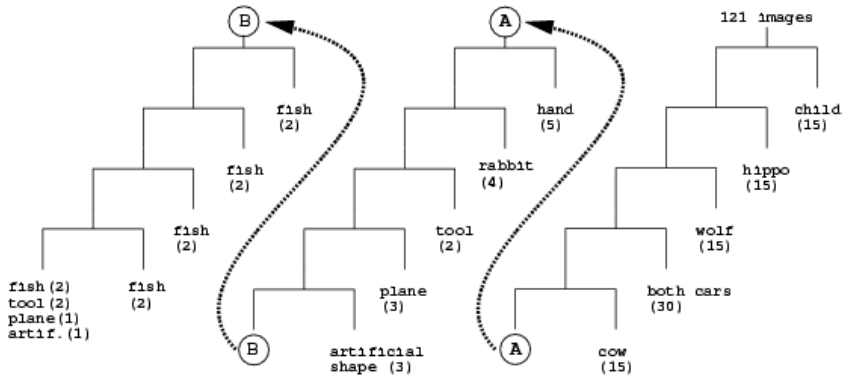


Figure 7.4: Factorization algorithm result. (From [30, p. 91])

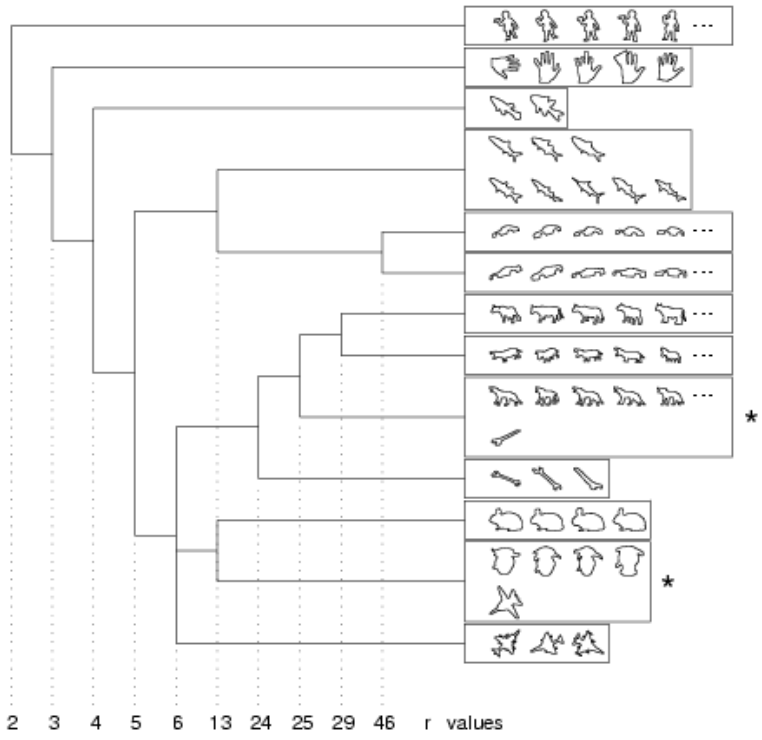


Figure 7.5: Typical cut algorithm result. (From [30, p. 90])

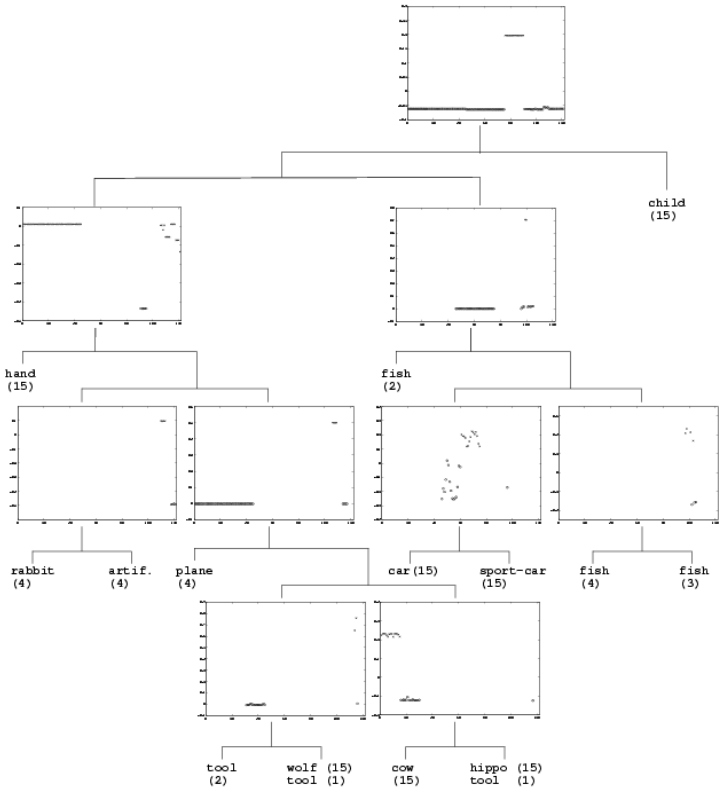


Figure 7.6: Normalized cut algorithm result. (From [30, p. 92])

mammal category, and then it separated quadrupeds from children and wrenches, which in turn formed their own cluster. However, an extra cluster was created which contained exemplars from the three quadruped classes. Finally, it correctly categorized the hands, the greebles, the rabbits, the planes, but it split the fish class into two components.

Our results compare favorably with those obtained using a recursive application of Perona and Freeman's factorization algorithm [91] (see Figure 7.4), which does not suggest a natural hierarchical interpretation of the data, and are comparable with the ones produced by typical/normalized cut algorithms [32, 112] (see Figures 7.5 and 7.6), some mistakes of which are similar to ours (e.g., mixing up wrenches with mammals at a coarse scale and splitting the fishes into two or more classes). At the base level, they misplaced one or more wrenches (both algorithms) and a plane (typical cut). The hierarchies produced by the three algorithms, however, are different, with ours being shallower.

Interestingly, as observed by Gdalyahu [30, 32], the data in this database form chained structures where images in the same class might be related to each other indirectly via "mediating" images. In fact, while the single-link algorithm succeeds to find meaningful structures, the complete-link approach fails completely on these data. In view of this, it is remarkable that our algorithm, which favors compact (although not necessarily isotropic) structures, produced reasonable results and succeeded in discovering meaningful hierarchical relations.

Chapter 8

Discussion and Conclusions

Our goal in this work was to apply cluster analysis as a unified approach for a wide range of vision applications. Many vision applications can be viewed as data partitioning problems, and can be addressed by this unified approach. In particular, we have focused on the tasks of image segmentation and image organization. In our approach, the visual objects (pixels, images) are mapped to the nodes of an undirected edge-weighted graph, with edge weights representing pairwise similarity relations. We use a canonical weight transformation, which converts the perceptual measure of distance to exponentially decaying similarity weight.

The clustering algorithm proposed in this thesis is centered around the novel notion of a *dominant set* (Section 2.1). Our proposal seems to be a plausible generalization of the notion of a maximal clique in the context of edge-weighted graphs since, in the unweighted case, dominant sets turn out to be equivalent to (strictly) maximal cliques (Section 2.4).

We also have established a correspondence between dominant sets and the extrema of a quadratic form over the standard simplex (Chapter 3). This is interesting because recently other quadratic programming formulations have been proposed for clustering and segmentation, though motivated by the different idea of finding cuts in a similarity graph [112, 44] or computing eigenvalues and eigenvectors of the weighted adjacency matrix [91, 104]. Computationally, this allows us to find clusters using straightforward continuous optimization techniques such as *payoff-monotonic game dynamics* (Chapter 4), a class of dynamical systems arising in evolutionary game theory. Such systems are attractive as can be coded in a few lines of any high-level programming language, can easily be implemented in a parallel network of locally interacting units, and offer the advantage of biological plausibility.

Our second contribution in this work is an hierarchical extension (Chapter 6) of our partitional clustering algorithm. To this end, we have considered a family of parameterized (continuous) quadratic programs (Section 6.1) and have studied the properties of its solutions as a function of its (nonnegative) parameter. When the regularization parameter is zero the local solutions correspond to dominant sets, and as it grows larger solutions corresponding to small clusters disappear. We have determined bounds on the regularization parameter which induce bounds on the size of the surviving clusters (Section 6.2). These properties have motivated our hierarchical clustering algorithm (Section 6.3), which is based on the idea of varying the regularization parameter in a principled way during the clustering process.

In Chapters 5 and 7 we investigated experimentally the plausibility of our approach. Some of our results were compared with other methods, primarily with the recently proposed normalized cut algorithm [112], the typical cut algorithm [32] and the Felzenszwalb-Huttenlocher segmentation algorithm [23]. To be concrete, we presented in Section 5.1 our segmentation results on images used also in [112, 32, 23]; we compared with the Luo et al. clustering algorithm [67] the performance on shape categorization in Section 5.2; and we compared with the typical cut algorithm and other methods the performance on image database organization in Chapter 7.

In all cases, the quality of the results provided by our algorithm looks comparable or better to the one of the other methods. There are, however, areas where there is scope for improvement. As far as the image segmentation in concerned, our algorithm was proven to report meaningful segmentation even after a few iterations of the replicator dynamical system. However, for large images it would require an awful amount of time, since the time complexity of each iteration is quadratic on the number of data items. In addition, our algorithm need to be fed with a complete graph and does not allow standard techniques for making it sparse. Possible ways to overcome these problems include the use of both faster dynamics (or even different optimization techniques) and sampling procedures.

Appendix A

A Long Proof

In Chapter 2 we gave a number of graph-theoretic notions in order to provide the definition of a dominant set. In Chapter 3 we established a correspondence between dominant sets and the extrema of a quadratic form over the standard simplex. To do so, we exploited various characterizations of the graph-theoretic notions in terms of determinants (Section 3.1). Here we provide proofs.

Lemma A.1 *Let $S = \{i_1, i_2, \ldots, i_m\} \subseteq V$ be a non-empty subset of vertices and, w.l.o.g., assume $i_1 < i_2 < \cdots < i_m$. Then, we have:*

1. $\mathrm{w}_S(i_h) = (-1)^m \det({}^hB_S)$, *for any* $i_h \in S$
2. $\sum_{j \in S} a_{ij} \mathrm{w}_S(j) = (-1)^{m+1} \det(A_S)$, *where i is an arbitrary element of S*
3. $\sum_{i \in S} \mathrm{a}wdeg_S(i) \mathrm{w}_S(i) = (-1)^{m+1} \det(A_S)$
4. *if* $m > 1$, $\mathrm{w}_S(i) = \sum_{j \in S \setminus \{i\}} (a_{ij} - a_{hj}) \mathrm{w}_{S \setminus \{i\}}(j)$, *where h is an arbitrary element of* $S \setminus \{i\}$
5. $\mathrm{W}(S) = (-1)^m \det(B_S)$.

Proof: To begin with, note that statement 2 implies statement 3. In fact:

$$\begin{aligned} \sum_{i \in S} \mathrm{a}wdeg_S(i) \mathrm{w}_S(i) &= \frac{1}{m} \sum_{i \in S} \sum_{j \in S} a_{ij} \mathrm{w}_S(i) \\ &= \frac{1}{m} m (-1)^{m+1} \det(A_S) \\ &= (-1)^{m+1} \det(A_S) \end{aligned}$$

We now proceed by induction on m to prove statement 1 and 2. Given a matrix C, we denote with $C_{i,j}$ the matrix resulting from the deletion of the i-th row and j-th column from C.

If $m = 1$ (and $S = \{i_1\}$) we have:

1. ${}^1B_S = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$, hence $\mathrm{w}_S(i_1) = 1 = (-1)^1 \det({}^1B_S)$.
2. $A_S = (0)$, therefore $\sum_{j \in S} a_{ij} \mathrm{w}_S(j) = 0 = (-1)^{1+1} \det(A_S)$.

If $m > 1$, then:

1. exploding the determinant of ${}^hB_S$ along the $h+1$-th column, we obtain:

$$\det({}^hB_S) = (-1)^h \det\left[({}^hB_S)_{1,h+1}\right].$$

Exploding the determinant at right term along the h-th row, we have:

$$\begin{aligned}
\det({}^hB_S) \;=\; & (-1)^h \{ \\
& (-1)^{h+1} \det(A_{S\setminus\{i_h\}}) + \\
& \sum_{i_k < i_h} (-1)^{h+k+1} a_{i_h i_k} \det\left\{ \left[\left({}^hB_S \right)_{1,h+1} \right]_{h,k+1} \right\} + \\
& \sum_{i_k > i_h} (-1)^{h+k} a_{i_h i_k} \det\left\{ \left[\left({}^hB_S \right)_{1,h+1} \right]_{h,k} \right\} \\
& \}
\end{aligned}$$

Exploding the determinant of ${}^kB_{S\setminus\{i_h\}}$ along the $k+1$-th column, it can be shown that, if $i_k < i_h$:

$$\det\left\{ \left[\left({}^hB_S \right)_{1,h+1} \right]_{h,k+1} \right\} = (-1)^{1+(k+1)} \det\left({}^kB_{S\setminus\{i_h\}} \right) .$$

Similarly, exploding the determinant of ${}^{k-1}B_{S\setminus\{i_h\}}$, it can be proved that, if $i_k > i_h$:

$$\det\left\{ \left[\left({}^hB_S \right)_{1,h+1} \right]_{h,k} \right\} = (-1)^{1+k} \det\left({}^{k-1}B_{S\setminus\{i_h\}} \right) .$$

From these facts and the inductive hypothesis on $S \setminus \{i_h\}$, we obtain:

$$\begin{aligned}
\det({}^{h}B_S) &= -\sum_{i_k<i_h} a_{i_h i_k} \det({}^{k}B_{S\setminus\{i_h\}}) && \text{(A.1)}\\
&\quad -\sum_{i_k>i_h} a_{i_h i_k} \det({}^{k-1}B_{S\setminus\{i_h\}}) && \text{(A.2)}\\
&\quad -\det(A_{S\setminus\{i_h\}}) && \text{(A.3)}\\
&= (-1)^m \left[\sum_{i_k\neq i_h} a_{i_h i_k} \mathrm{w}_{S\setminus\{i_h\}}(i_k) - \sum_{i_k\neq i_h} a_{i_r i_k} \mathrm{w}_{S\setminus\{i_h\}}(i_k)\right] && \text{(A.4)}
\end{aligned}$$

for all $i_r \in S \setminus \{i_h\}$. Now, since statement 2 implies statement 3, we have:

$$\begin{aligned}
\det({}^{h}B_S) &= (-1)^m \left[\sum_{i_k\neq i_h} a_{i_h i_k} \mathrm{w}_{S\setminus\{i_h\}}(i_k) - \sum_{i_k\neq i_h} \mathrm{a}wdeg_{S\setminus\{i_h\}}(i_k)\mathrm{w}_{S\setminus\{i_h\}}(i_k)\right]\\
&= (-1)^m \sum_{i_k\neq i_h} \left[a_{i_h i_k} - \mathrm{a}wdeg_{S\setminus\{i_h\}}(i_k)\right] \mathrm{w}_{S\setminus\{i_h\}}(i_k)\\
&= (-1)^m \sum_{i_k\neq i_h} \phi_{S\setminus\{i_h\}}(i_k, i_h)\mathrm{w}_{S\setminus\{i_h\}}(i_k)\\
&= (-1)^m \mathrm{w}_S(i_h)
\end{aligned}$$

Note also that, from Equation A.4:

$$\det({}^{h}B_S) = (-1)^m \left[\sum_{i_k\neq i_h} (a_{i_h i_k} - a_{i_r i_k})\mathrm{w}_{S\setminus\{i_h\}}(i_k)\right]$$

which proves 4.

2. To conclude the proof, we need an additional notation. With $\left({}^{h}B_S\right)_{i,j;a_{rs}}$ we denote the matrix $\left[\left({}^{h}B_S\right)_{i,j}\right]_{p,q}$ where p, q are the row and column of the element a_{rs} in $\left({}^{h}B_S\right)_{i,j}$ respectively.

   Exploiting statement 1, we have for any $i_r \in S$:

$$\sum_{i_h\in S} a_{i_r i_h} w_S(i_h) \quad =$$

$$
\begin{aligned}
&= \sum_{i_h \in S} a_{i_r i_h} (-1)^m \det\left({}^h B_S\right) \\
&= (-1)^m \sum_{i_h \in S} a_{i_r i_h} \det\left({}^h B_S\right) \\
&= (-1)^m \sum_{i_h \in S} a_{i_r i_h} (-1)^h [\\
&\qquad\qquad (-1)^{r+1} \det\left[(A_S)_{r,h}\right] + \\
&\qquad\qquad \sum_{i_k < i_h} (-1)^{r+k+1} a_{i_r i_k} \det\left[({}^h B_S)_{1,h+1;a_{i_r i_k}}\right] + \\
&\qquad\qquad \sum_{i_k > i_h} (-1)^{r+k} a_{i_r i_k} \det\left[({}^h B_S)_{1,h+1;a_{i_r i_k}}\right] \\
&\qquad\qquad] \\
&= (-1)^{m+1} \det\left(A_S\right) + \\
&\quad +(-1)^m \sum_{i_h \in S} [\\
&\qquad\qquad \sum_{i_k < i_h} (-1)^{r+k+1+h} a_{i_r i_h} a_{i_r i_k} \det\left[({}^h B_S)_{1,h+1;a_{i_r i_k}}\right] + \\
&\qquad\qquad \sum_{i_k > i_h} (-1)^{r+k+h} a_{i_r i_h} a_{i_r i_k} \det\left[({}^h B_S)_{1,h+1;a_{i_r i_k}}\right] \\
&\qquad\qquad] \\
&= (-1)^{m+1} \det\left(A_S\right) + \\
&\quad +(-1)^m [\\
&\qquad\qquad \sum_{i_k < i_h} (-1)^{r+k+1+h} a_{i_r i_h} a_{i_r i_k} \det\left[({}^h B_S)_{1,h+1;a_{i_r i_k}}\right] + \\
&\qquad\qquad \sum_{i_h < i_k} (-1)^{r+k+h} a_{i_r i_h} a_{i_r i_k} \det\left[({}^h B_S)_{1,h+1;a_{i_r i_k}}\right] \\
&\qquad\qquad] \\
&= (-1)^{m+1} \det\left(A_S\right) + \\
&\quad +(-1)^m \{ \sum_{i_k < i_h} [\\
&\qquad\qquad\qquad (-1)^{r+k+1+h} a_{i_r i_h} a_{i_r i_k} \det\left[({}^h B_S)_{1,h+1;a_{i_r i_k}}\right] + \\
&\qquad\qquad\qquad (-1)^{r+h+k} a_{i_r i_k} a_{i_r i_h} \det\left[({}^h B_S)_{1,k+1;a_{i_r i_h}}\right] \\
&\qquad\qquad\qquad]\} \\
&= (-1)^{m+1} \det\left(A_S\right)
\end{aligned}
$$

The last equality arises from the fact that if $i_p, i_q \in S$ with $i_p \neq i_q$, then we have $({}^pB_S)_{1,p;a_{ii_q}} = ({}^pB_S)_{1,q;a_{ii_p}}$ for any $i \in S$.

Finally, to prove statement 5, we explode the determinant of B_S along the first line:

$$\begin{aligned}\det(B_S) &= \sum_{j=1}^{m}(-1)^{1+(j+1)}\det\left[(B_S)_{1,j+1}\right] \\ &= \sum_{j=1}^{m}(-1)^{j}(-1)^{1+(j+1)}\det\left({}^jB_S\right)\end{aligned}$$

where, in the last equality, we have exploded the determinant of ${}^jB_S$ along the $j+1$-th column. Now it follows easily that:

$$\det(B_S) = \sum_{j=1}^{m}\det\left({}^jB_S\right) = (-1)^m\sum_{j\in S}\mathrm{w}_S(j)$$

□

References

[1] M. Abdel-Mottaleb, S. Krishnamachari, and N.Mankovich. Performance evaluation of clustering algorithms for scalable image retrieval. In *Proc. IEEE Workshop on Empirical Evaluation Techniques in Computer Vision*, 1998.

[2] L. Angelini, L. Nitti, M. Pellicoro, and S. Stramaglia. Cost functions for pairwise data clustering. *Physics Letters A*, 285:279–285, 2001.

[3] J. G. Auguston and J. Minker. An analysis of some graph theoretical clustering techniques. *J. ACM*, 17(4):571–588, 1970.

[4] K. Barnard, P. Duygulu, and D. Forsyth. Clustering art. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2001.

[5] K. Barnard and D. Forsyth. Learning the semantics of words and pictures. In *Proc. IEEE Int. Conf. Computer Vision*, volume 2, pages 408–415, 2001.

[6] L. E. Baum and J. A. Eagon. An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology. *Bull. Amer. Math. Soc.*, 73:360–363, 1967.

[7] A. Del Bimbo. *Visual information retrieval.* San Francisco, 1999.

[8] C. M. Bishop, M. Svensn, and C. K. I. Williams. Gtm: The generative topographic mapping. *Neural Comp.*, 10:215–234, 1998.

[9] A. Blake and A. Zisserman. *Visual Recognition.* MIT Press, Cambridge, MA, 1987.

[10] M. Blatt, S. Wiseman, and E. Domany. Data clustering using a model granular magnet. *Neural Computation*, 9:1805–1842, 1997.

[11] I. M. Bomze. Evolution towards the maximum clique. *J. Global Optim.*, 10:143–164, 1997.

[12] I. M. Bomze, M. Budinich, P. M. Pardalos, and M. Pelillo. The maximum clique problem. In D.-Z. Du and P. M. Pardalos, editors, *Handbook of Combinatorial Optimization (Suppl. Vol. A)*, pages 1–74. Kluwer, Boston, MA, 1999.

[13] J. Buhmann. The handbook of brain theory and neural networks. In M.A. Arbib, editor, *Data Clustering and Learning*. The MIT Press, Cambridge, MA, 2002. Second edition.

[14] C. Carson, S. Belongie, H. Greenspan, and J. Malik. Region-based image querying. In *Proc. Proc. of the IEEE Workshop on Content-based Access of Image and Video libraries*, pages 42–49, 1997.

[15] K. I. Chang, K. W. Bowyer, and M. Sivagurunath. Evaluation of texture segmentation algorithms. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 1063–1069, 1999.

[16] J. Chen, C.A. Bouman, and J.C. Dalton. Hierarchical browsing and search of large image databases. *IEEE transactions on Image Processing*, 9(3):442–455, 2000.

[17] K. Cho and P. Meer. Image segmentation from consensus information. *Computer Vision and Image Understanding*, 68:72–89, 1997.

[18] J. Costeira and T. Kanade. A multibody factor- ization method for motion analysis. In *In Proc. In- ternational Conf. Computer Vision*, pages 1071–1076, 1995.

[19] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *J. R. Statist. Soc. B*, 39:1–38, 1977.

[20] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* J. Wiley & Sons, New York, 1973.

[21] D. Ennis. Modeling similarity and identification when there are momentary fluctuations in psychological amplitudes. In F. Ashby, editor, *Multidimensional Models of Perception and Cognition*, pages 279–298. Lawrence Erlbaum assoc. publishers, 1992.

[22] B. S. Everitt. *Cluster Analysis*. J. Wiley & Sons, New York, 1974.

[23] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. Preprint available at http://www.cs.cornell.edu/∼dph/papers/seg-revised.pdf.

[24] B. Fischer and J. M. Buhmann. Path-based clustering for grouping smooth curves and texture segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(4):513–518, 2003.

[25] R. A. Fisher. *The Genetical Theory of Natural Selection*. Oxford University Press, London, UK, 1930.

[26] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. *Computer Graphics Principles and Practice.* Addison-Wesley, Reading, MA, 1993.

[27] D. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 2002.

[28] D. Fudenberg and D. K. Levine. *The Theory of Learning in Games*. MIT Press, Cambridge, MA, 1998.

[29] K. Fukunaga. *Introduction to statistical pattern recognition.* Academic Press, San Diego, 1990.

[30] Y. Gdalyahu. *Stochastic Clustering and its Applications to Computer Vision*. PhD thesis, Hebrew University, Jerusalem, Israel, 1999.

[31] Y. Gdalyahu and D. Weinshall. Flexible syntactic matching of curves and its application to automatic hierarchical classification of silhouettes. *IEEE Trans. Pattern Anal. Machine Intell.*, 21(12):1312–1328, 1999.

[32] Y. Gdalyahu, D. Weinshall, and M. Werman. Sef-organization in vision: Stochastic clustering for image segmentation, perceptual grouping, and image database organization. *IEEE Trans. Pattern Anal. Machine Intell.*, 23(10):1053–1074, 2001.

[33] C. W. Gear. Feature grouping in moving images. In *In Proc, IEEE workshop on motion of non-rigid and articulated objects*, pages 214–219, 1994.

[34] N. Geller. On the citation influence methodology of pinski and narin. *Information Processing and Management*, pages 93–95, 1978.

[35] D. Geman, S. Geman, C. Graffigne, and P. Dong. Boundary detection by constrained optimization. *IEEE Trans. Pattern Anal. Machine Intell.*, 12(7):609–628, 1990.

[36] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Trans. Pattern Anal. Machine Intell.*, 6:721–741, 1984.

[37] L. E. Gibbons, D. W. Hearn, P. M. Pardalos, and M. V. Ramana. Continuous characterizations of the maximum clique problem. *Math. Oper. Res.*, 22:754–768, 1997.

[38] J. Goldberger, H. Greenspan, and S. Gordon. Unsupervised image clustering using the information bottleneck method. In *Proc. DAGM*, 2002.

[39] S. Gordon, H. Greenspan, and J. Goldberger. Applying the information bottleneck principle to unsupervised clustering of discrete and continuous image representations. In *Proc. IEEE Int. Conf. Computer Vision*, 2003.

[40] C. C. Gotlieb and S. Kumar. Semantic clustering of index terms. *J. ACM*, 15(4):493–513, 1968.

[41] T. Graepel. Statistical physics of clustering algorithms. Diplomarbeit, Technique Universitt, FB Physik, Institut fr Theoretishe Physik, Berlin, 1998.

[42] H. Greenspan, J. Goldberger, and L. Ridel. A continuous probabilistic framework for image matching. *Journal of Computer Vision and Image Understanding*, 84:384–406, 2001.

[43] H. Greenspan, S. Gordon, and J. Goldberger. Probabilistic models for generating, mod- elling and matching image categories. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2002.

[44] W. W. Hager and Y. Krylyuk. Graph partitioning and continuous quadratic programming. *SIAM Journal on Discrete Mathematics*, 12:500–523, 1999.

[45] T. Hastie and W. Stuetzle. Principal curves. *Journal of the American Statistical Associa tion*, 84:502–516, 1989.

[46] L. Herault and R. Horaud. Figure-ground discrimination: A combinatorial optimization approach. *IEEE Trans. Pattern Anal. Machine Intell.*, 15(9):899–914, 1993.

[47] J. Hofbauer. Imitation dynamics for games. Collegium Budapest, preprint, 1995.

[48] J. Hofbauer and K. Sigmund. *Evolutionary Games and Population Dynamics*. Cambridge University Press, Cambridge, UK, 1998.

[49] T. Hofmann and J. Buhmann. Pairwise data clustering by deterministic annealing. *IEEE Trans. Pattern Anal. Machine Intell.*, 19(1):1–14, 1997.

[50] T. Hofmann, J. Puzicha, and J. Buhmann. Unsupervized texture segmentation in a deterministic annealing framework. *IEEE Trans. Pattern Anal. Machine Intell.*, 20:803–818, 1998.

[51] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, Cambridge, UK, 1985.

[52] J. Huang, S. R. Kumar, and R. Zabith. An automatic hierarchical image classification scheme. In *Proc. ACM Conference on multimedia*, pages 219–228, 1998.

[53] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, Englewood Cliffs, NJ, 1988.

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

[55] E. Johnson, A. Mehrotra, and G. Nemhauser. Mincut clustering. *Mathematical program ming*, 62:133–151, 1993.

[56] W. Johnson and J. Lindenstrauss. Extension of lipshitz mapping to hilbert space. *Contemp. Math.*, 26:189–206, 1984.

[57] J.W. Sammon Jr. A non-linear mapping for data structure analysis. *IEEE Trans. Computers*, 18:401–409, 1969.

[58] D. Karger and C. Stein. A new approach to the minimum cut problem. *Journal of the ACM*, 43:601–640, 1996.

[59] P. N. Klein, T. B. Sebastian, and B. B. Kimia. Shape matching using edit-distance: An implementation. In *Proc. 11th Ann. ACM-SIAM Symp. Discrete Algorithms (SODA)*, pages 781–190, 2001.

[60] J. Kleinberg. Authoritative sources in a hyperlink environment. In *Proc. 9th ACM-SIAM Symposium on Discrete Algorithms*, pages 668–677, 1998.

[61] H. Klock and J. Buhmann. Multidimensional scaling by deterministic annealing. In *Energy Minimization Methods in Computer Vision and Pattern Recognition*, Lecture Notes in Computer Science, pages 245–260. Springer-Verlag, Berlin, 1997.

[62] S. Krishnamachari and M. Abdel-Mottaleb. Hierarchical clustering algorithm for fast image retrieval. In *Proc. SPIE Conference on Storage and Retrieval for Image and Video databases VII*, pages 427–435, 1999.

[63] J. B. Kruskal. Nonmetric multidimensional scaling: a numerical method. *Psychometrika*, 29:115–129, 1964.

[64] G. Lance and W. Williams. A general theory of classification sorting strategies: Ii. clustering systems. *Computer Journal*, 10:271–277, 1969.

[65] V. Losert and E. Akin. Dynamics of games and genes: Discrete versus continuous time. *J. Math. Biol.*, 17:241–251, 1983.

[66] D. G. Luenberger. *Linear and Nonlinear Programming.* Addison-Wesley, Reading, MA, 1984.

[67] B. Luo, A. Robles-Kelly, A. Torsello, R. C. Wilson, and E. R. Hancock. A probabilistic framework for graph clustering. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2001.

[68] S. P. Luttrel. *Neural Computation*, 6:767, 1994.

[69] Yu. Lyubich, G. D. Maistrowskii, and Yu. G. Ol'khovskii. Selection-induced convergence to equilibrium in a single-locus autosomal population. *Problems of Information Transmission*, 16:66–75, 1980.

[70] J. Malik and P. Perona. Preattentive texture discrimination with early vision mechanisms. *J. Optical Soc. Am.*, 7(2):923–932, 1990.

[71] Jitendra Malik, Serge Belongie, Thomas Leung, and Jianbo Shi. Contour and texture analysis for image segmentation. *International Journal of Computer Vision*, 43(1):7–27, 2001.

[72] J. Maynard Smith. *Evolution and the Theory of Games*. Cambridge University Press, Cambridge, UK, 1982.

[73] D. Miller and K. Rose. Hierarchical, unsupervised learning with growing via phase transitions. *Neural Computation*, 8:425–450, 1996.

[74] T. S. Motzkin and E. G. Straus. Maxima for graphs and a new proof of a theorem of Turán. *Canad. J. Math.*, 17:533–540, 1965.

[75] D. Mumford and J. Shah. Boundary detection by minimizing functionals. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 22–26, 1985.

[76] H. Nagamochi and T. Ibaraki. Computing edge connectivity in multigraphs and capacitated graphs. *SIAM Journal of Discrete Mathematics*, 5:54–66, 1992.

[77] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS*, 2001.

[78] N. Pal and S. Pal. A review on image segmentation techniques. *Pattern Recognition*, 26:1277–1294, 1993.

[79] M. Pavan and M. Pelillo. Dominant sets and hierarhical clustering. In *Proc. IEEE Int. Conf. Computer Vision*, volume 1, pages 362–369, 2003.

[80] M. Pavan and M. Pelillo. Generalizing the motzkin-straus theorem to edge-weighted graphs, with applications to image segmentation. In M. Figueiredo, A. Rangarajan, and J. Zerubia, editors, *Energy Minimization Methods in Computer Vision and Pattern Recognition*, volume 2683 of *Lecture Notes in Computer Science*, pages 485–500. Springer-Verlag, Berlin, 2003.

[81] M. Pavan and M. Pelillo. Image segmentation by dominant sets. In A. Del Lungo, V. Di Gesù, and A. Kuba, editors, *International Workshop on Combinatorial Image Analysis*, Electronic Notes in Discrete Mathematics. Elsevier Science, 2003.

[82] M. Pavan and M. Pelillo. A new graph-theoretic approach to clustering and segmentation. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, volume 1, pages 145–152, 2003.

[83] M. Pavan and M. Pelillo. Pairwise data clustering using monotone game dynamics. In A. Cappelli and F. Turini, editors, *National Congress of Associazione Italiana per l'Intelligenza Artificiale*, Lecture Notes in Computer Science. Springer-Verlag, Berlin, 2003.

[84] M. Pavan and M. Pelillo. Unsupervised texture segmentation by dominant sets and game dynamics. In *Proc. IEEE Int. Conf. on Image Analysis and Processing*, pages 302–307, 2003.

[85] M. Pelillo. The dynamics of nonlinear relaxation labeling processes. *J. Math. Imaging Vision*, 7(4):309–323, 1997.

[86] M. Pelillo. Replicator equations, maximal cliques, and graph isomorphism. *Neural Computation*, 11(8):2023–2045, 1999.

[87] M. Pelillo. Matching free trees, maximal cliques, and monotone game dynamics. *IEEE Trans. Pattern Anal. Machine Intell.*, 24(11):1535–1541, 2002.

[88] M. Pelillo and A. Jagota. Feasible and infeasible maxima in a quadratic program for maximum clique. *J. Artif. Neural Networks*, 2:411–420, 1995.

[89] M. Pelillo, K. Siddiqi, and S. W. Zucker. Matching hierarchical structures using association graphs. *IEEE Trans. Pattern Anal. Machince Intell.*, 21(11):1105–1120, 1999.

[90] F. Pereira, N. Tishby, and L. Lee. Distributional clustering of english words. In *30th Annual Meeting of the Association for Computational Linguistics*, pages 183–190, 1993.

[91] P. Perona and W. Freeman. A factorization approach to grouping. In H. Burkhardt and B. Neumann, editors, *Computer Vision—ECCV'98*, pages 655–670. Springer-Verlag, Berlin, 1998.

[92] G. Pinski and F. Narin. Citation influence for journal aggregates of scientific publications: theory with application to the literature of physics. *Information Processing and Management*, 12:297–312, 1976.

[93] A. Prelic, M. Pavan, and M. Pelillo. Matching of modified x-y trees for document classification. In *Proc. International Workshop on Artificial Neural Networks in Pattern Recognition*, 2003.

[94] J. Puzicha, T. Hofmann, and J. M. Buhmann. Histogram clustering for unsupervised segmentation and image retrieval. *Pattern Recognition Letters*, 20:899–909, 1999.

[95] J. Puzicha, T. Hofmann, and J. M. Buhmann. A theory of proximity based clustering: Structure detection by optimization. *Pattern Recognition*, 33(4):617–634, 1999.

[96] V. V. Raghavan and C. T. Yu. A comparison of the stability characteristics of some graph theoretic clustering methods. *IEEE Trans. Pattern Anal. Machine Intell.*, 3:393–402, 1981.

[97] A. Robles-Kelly and Edwin R. Hancock. An expectationmaximisation framework for segmentation and grouping. *Image and Vision Computing*, 20:725–738, 2002.

[98] A. Robles-Kelly, S. Sarkar, and E. R. Hancock. A fast leading eigenvector approximation for segmentation and grouping. In *Proc. 16th International Conference on Pattern Recognition*, volume 2, pages 639–6422. IEEE Computer Society Press, 2002.

[99] E. Rosch and C. Mervis. Family resemblance: studies in the internal structure of categories. *Cognitive Psychology*, 7:573–605, 1975.

[100] K. Rose. Deterministic annealing for clustering, compression, classification, regression, and related optimization problems. In *Proceedings of the IEEE*, pages 2210–2238, 1998.

[101] A. Rosenfeld, R. A. Hummel, and S. W. Zucker. Scene labeling by relaxation operations. *IEEE Trans. Syst. Man & Cybern.*, 6:420–433, 1976.

[102] Y. Rubner, J. Puzicha, Carlo Tomasi, and J. M. Buhmann. Empirical evaluation of dissimilarity measures for color and texture. *Computer Vision and Image Understanding*, 84:25–43, 2001.

[103] L. Samuelson. *Evolutionary Games and Equilibrium Selection*. MIT Press, Cambridge, MA, 1997.

[104] S. Sarkar and K. L. Boyer. Quantitative measures of change based on feature organization: Eigenvalues and eigenvectors. *Computer Vision and Image Understanding*, 71(1):110–136, 1998.

[105] G. L. Scott and H. C. Longuet-Higgins. Feature grouping by relocalisation of eigenvectors of the proxmity matrix. In *In Proc. British Machine Vi- sion Conference*, pages 103–108, 1990.

[106] K. Sengupta and K. L. Boyer. Organizing large structural modelbases. *IEEE Trans. Pattern Anal. Machine Intell.*, 17(4):321–332, 1995.

[107] E. Sharon, A. Brandt, and R. Basri. Fast multiscale image segmentation. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, volume 1, pages 70–77, 2000.

[108] G. Sheikholeslami and A. Zhang. Approach to clustering large visual databases using wavelet transform. In *Proc. SPIE conference on visual data exploration and analysis IV*, volume 3017, 1997.

[109] R. Shepard. Multidimensional scaling, tree fitting, and clustering. *Science*, 210:390–398, 1980.

[110] R. Shepard. Toward a universal law of generalization for psychological science. *Science*, 237:1317–1323, 1987.

[111] R. Shepard and P. Arabie. Additive clustering: Representation of similarities as combinations of discrete overlapping properties. *Psychological Review*, 86:87–123, 1979.

[112] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Machine Intell.*, 22(8):888–905, 2000.

[113] K. Siddiqi, A. Shokoufaundeh, S. J. Dickinson, and S. W. Zucker. Shock graphs and shape matching. *Int. J. Computer Vision*, 35(1):1999, 1999.

[114] V. T. Sós and E. G. Straus. Extremal of functions on graphs with applications to graphs and hypergraphs. *J. Combin. Theory B*, 32:246–257, 1982.

[115] A. Torsello and M. Pelillo. Continuous-time relaxation labeling processes. *Pattern Recognition*, 33:1897–1908, 2000.

[116] A. Tversky. Features of similarity. *Psychological Review*, 84:327–352, 1977.

[117] J. W. Weibull. *Evolutionary Game Theory*. MIT Press, Cambridge, MA, 1995.

[118] Y. Weiss. Segmentation using eigenvectors: A unifying view. In *Proc. Int. Conf. Computer Vision*, pages 975–979, 1999.

[119] H. S. Wilf. Spectral bounds for the clique and independence numbers of graphs. *J. Combin. Theory B*, 40:113–117, 1986.

[120] Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE Trans. Pattern Anal. Machine Intell.*, 15(11):1101–1113, 1993.

[121] C. T. Zahn. Graph-theoretic methods for detecting and describing gestalt clusters. *IEEE Trans. Comput.*, 20:68–86, 1971.

[122] T. Zoller and J. M. Buhmann. Active learning for hierarchical pairwise data clustering. In *Proc. of the International Conference on Pattern Recognition*, volume 2, pages 186–189, 2000.

[123] S. W. Zucker, A. Dobbins, and L. Iverson. Two stages of curve detection suggest two styles of visual computation. *Neural Computation*, 1:68–81, 1989.