Dominant Sets and Hierarchical Clustering

Massimiliano Pavan and Marcello Pelillo
Dipartimento di Informatica
Università Ca' Foscari di Venezia
Via Torino 155, 30172 Venezia Mestre, Italy
{mapavan, pelillo}@dsi.unive.it

Abstract

Dominant sets are a new graph-theoretic concept that has proven to be relevant in partitional (flat) clustering as well as image segmentation problems. However, in many computer vision applications, such as the organization of an image database, it is important to provide the data to be clustered with a hierarchical organization, and it is not clear how to do this within the dominant set framework. In this paper we address precisely this problem, and present a simple and elegant solution to it. 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. Straightforward dynamics from evolutionary game theory are used to locate the solutions of the quadratic programs at each level of the hierarchy. We apply the proposed framework to the problem of organizing a shape database. Experiments with three different similarity matrices (and databases) reported in the literature have been conducted, and the results confirm the effectiveness of our approach.

1. Introduction

The unsupervised partitioning of data (or clustering) is a problem that pervades computer vision research, and recently there has been a resurgence of interest around graph-based (pairwise) approaches [1, 6, 18, 20, 22], which treat the data to be clustered (pixels, edge elements, etc.) as vertices of a *similarity* (edge-weighted) graph, where the edges represent neighborhood relations, and the weights reflect the similarity between data. Graph-theoretic clustering algorithms basically consist of searching for certain combinatorial structures in the similarity graph, such as a minimum spanning tree [26] or a minimum cut [6, 22, 25] and, among these methods, a classic approach (the "complete-link" algorithm [8]) reduces to a search for a complete subgraph, namely a *clique*.[1] Indeed, some authors [2, 19] argue that the maximal clique is the strictest definition of a cluster.

In a recent paper [14], we have developed a new framework for partitional (i.e., flat) 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. An intriguing connection between dominant sets and the solutions of a (continuous) quadratic optimization problem allows the use of straightforward dynamics from evolutionary game theory to determine them [24]. The approach has proven to be a powerful one when applied to problems such as intensity, color, and texture segmentation [14, 15].

However, in many computer vision applications, such as the organization of an image database [21], it is important to organize the data to be clustered in a hierarchical manner, and it is not obvious how to do this within the dominant set framework. In this paper we address precisely this problem, and provide a simple and elegant solution to it.

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

[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.

0-7695-1950-4/03 $17.00 © 2003 **IEEE**

IEEE COMPUTER SOCIETY

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 simple replicator dynamics are used as local optimization procedures.

We demonstrate the potential of our framework to the problem of organizing a shape database. Experiments with various similarity matrices (and databases) reported in the literature have been conducted, and the results confirm the power of the approach in correctly partitioning the data as well as discovering meaningful hierarchial relations, despite its simplicity.

2. Dominant sets and their characterization

2.1. The notion of a dominant set

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 $w : E \to \mathbb{R}_+^*$ 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 $n \times n$ nonnegative, symmetric matrix $A = (a_{ij})$ defined as:

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

Clearly, since there are no self-loops, all the elements on the main diagonal of A are zero.

Intuitively, a cluster should satisfy two fundamental conditions: it should have high internal homogeneity, and 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 weights on the vertices. This perspective gives us a chance to analyze the assignment of the edge-weights in a simpler and fruitful way.

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:

$$\mathrm{awdeg}_S\,(i) = \frac{1}{|S|} \sum_{j \in S} a_{ij}\ . \tag{1}$$

Observe that $\mathrm{awdeg}_{\{i\}}\,(i) = 0$ for any $i \in V$. Moreover, if $j \notin S$ we define:

$$\phi_S\,(i,j) = a_{ij} - \mathrm{awdeg}_S\,(i)\ . \tag{2}$$

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.

We are now in a position to formalize the notion of "induction" of node-weights, which is captured by the following recursive definition. Let $S \subseteq V$ be a non-empty subset of vertices and $i \in S$. The *weight* of i w.r.t. S is

$$\mathrm{w}_S\,(i) = \begin{cases} 1, & \text{if } |S| = 1 \\ \displaystyle\sum_{j \in S \setminus \{i\}} \phi_{S \setminus \{i\}}\,(j,i)\, \mathrm{w}_{S \setminus \{i\}}\,(j)\ , & \text{otherwise.} \end{cases} \tag{3}$$

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

$$\mathrm{W}(S) = \sum_{i \in S} \mathrm{w}_S(i)\ . \tag{4}$$

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. 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 $S \setminus \{i\}$.

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

Definition 1 *A non-empty subset of vertices $S \subseteq V$ such that $W(T) > 0$ 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 $W(T) > 0$ for any non-empty $T \subseteq S$ is a technicality explained in some detail in [16].

It turns out that, when applied to unweighted graphs, the notion of a dominant set coincides with that of a (strictly) maximal clique [16].

2.2. Dominant sets as local maxima

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 [13]):

$$\begin{array}{ll} \text{maximize} & f(\mathbf{x}) = \mathbf{x}'A\mathbf{x} \\ \text{subject to} & \mathbf{x} \in \Delta \end{array} \tag{5}$$

where

$$\Delta = \{\mathbf{x} \in \mathbb{R}^n \ : \ x_i \geq 0 \text{ for all } i \in V \text{ and } \mathbf{e}'\mathbf{x} = 1\}$$

is the standard simplex of $\mathbb{R}^n$, $\mathbf{e}$ is a vector of appropriate length consisting of unit entries (hence $\mathbf{e}'\mathbf{x} = \sum_i x_i$), and a prime denotes transposition.

The *support* of a vector $\mathbf{x} \in \Delta$ is defined as the set of indices corresponding to its positive components, that is:

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

The following theorem, proved in [14], establishes an intriguing connection between dominant sets and local solutions of program (5).

Theorem 1 *If S is a dominant subset of vertices, then its weighted characteristics vector $\mathbf{x}^S$, which is the vector of Δ defined as*

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

is a strict local solution of program (5).

Conversely, if $\mathbf{x}$ is a strict local solution of program (5) then its support $S = \sigma(\mathbf{x})$ is a dominant set, provided that $\mathrm{w}_{S\cup\{i\}}(i) \neq 0$ for all $i \notin S$.

The condition that $\mathrm{w}_{S\cup\{i\}}(i) \neq 0$ for all $i \notin S = \sigma(\mathbf{x})$ is a technicality explained in [16].

By virtue of this result, we can find a dominant set by first localizing a solution of program (5) with an appropriate continuous optimization technique, and then picking up the support set of the solution found. In this sense, we indirectly perform combinatorial optimization via continuous optimization. Note that the components of the weighted characteristic vectors give us a measure of the participation of the corresponding vertices in the cluster, whereas the value of the objective function measures the cohesiveness of the class.

3. 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:

$$\begin{array}{ll} \text{maximize} & f_\alpha(\mathbf{x}) = \mathbf{x}'(A - \alpha I)\mathbf{x} \\ \text{subject to} & \mathbf{x} \in \Delta \end{array} \tag{6}$$

where $\alpha \geq 0$ is a parameter and I is the identity matrix, which includes as special case program (5) when $\alpha = 0$.

Note that the solutions of (6) remain the same if the matrix $A - \alpha I$ is replaced with $A - \alpha I + \kappa \mathbf{e}\mathbf{e}'$, where κ is an arbitrary parameter, since $\mathbf{x}'(A - \alpha I + \kappa\mathbf{e}\mathbf{e}')\mathbf{x} = \mathbf{x}'(A - \alpha I)\mathbf{x} + \kappa$ for all $\mathbf{x} \in \Delta$. In particular, if $\kappa = \alpha$ the resulting matrix is nonnegative and has a null diagonal. Hence, Theorem 1 applies and all (strict) solutions of (6) correspond to dominant sets for the scaled similarity matrix $A + \alpha(\mathbf{e}\mathbf{e}' - I)$ having the off-diagonal entries equal to $a_{ij} + \alpha$.

A point $\mathbf{x} \in \Delta$ satisfies the Karush-Kuhn-Tucker (KKT) conditions for problem (6), i.e. the first-order necessary conditions for local optimality [10], 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 $\lambda = \mathbf{x}'(A - \alpha I)\mathbf{x}$. Hence the previous conditions can be explicitly rewritten as

$$\begin{array}{lcl} (A\mathbf{x})_i - \alpha x_i & = & \mathbf{x}'A\mathbf{x} - \alpha\mathbf{x}'\mathbf{x}, \text{ if } i \in \sigma(\mathbf{x}) \\ (A\mathbf{x})_i & \leq & \mathbf{x}'A\mathbf{x} - \alpha\mathbf{x}'\mathbf{x}, \text{ otherwise} \end{array} \tag{7}$$

A straightforward way to find (local) solutions of program (6) is given by the so-called *replicator dynamics*, a class of continuous- and discrete-time dynamical systems arising in evolutionary game theory, which are also intimately related to relaxation labeling processes [12]. Such systems are attractive because they 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.

In our simulations, we used the following model

$$x_i(t+1) = x_i(t)\frac{(A\mathbf{x})_i - \alpha x_i(t)}{\mathbf{x}(t)'(A - \alpha I)\mathbf{x}(t)} \tag{8}$$

for $i = 1 \ldots n$, which corresponds to the discrete-time version of first-order replicator equations (see, e.g., [24]).

Provided that the matrix $A - \alpha I$ is scaled properly to avoid negative values, it is readily seen that the simplex Δ is invariant under these dynamics, which means that every trajectory starting in Δ will remain in Δ for all future times. Moreover, it can be proven that, since A is symmetric, the (scaled) objective function f_α is strictly increasing along any nonconstant trajectory of (8), and its asymptotically stable points are in one-to-one correspondence to strict local solutions of (6) [24]. These, in turn, correspond to dominant sets for the scaled similarity matrix $A + \alpha(\mathbf{e}\mathbf{e}' - I)$. These properties naturally suggest using replicator equations as a useful heuristic for finding local solutions of (6) and hence dominant sets.

4. Bounds for the regularization parameter

The objective function f_α in (6) consists of a data term and a regularization term. The first one ($\mathbf{x}'A\mathbf{x}$) favors solutions with high internal coherency, and the second ($-\alpha\mathbf{x}'\mathbf{x}$), 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) 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, we introduce the following additional notations. Given a subset of vertices $S \subseteq V$, the face of Δ corresponding to S 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, $\Delta_V = \Delta$ and, accordingly, we shall write $\mathrm{int}(\Delta)$ instead of $\mathrm{int}(\Delta_V)$.

Also, 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 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) 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) 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 (7) we have:

$$(A\mathbf{x})_i \leq \mathbf{x}'A\mathbf{x} - \alpha\mathbf{x}'\mathbf{x} < 0$$

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 2 *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

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

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:

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

where the first inequality follows from the Rayleigh-Ritz theorem [7], 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

$$-\alpha < \frac{(A\mathbf{x})_i - \mathbf{x}'A\mathbf{x}}{\mathbf{x}'\mathbf{x}}$$

which yields $\mathbf{x}'A\mathbf{x} - \alpha\mathbf{x}'\mathbf{x} < (A\mathbf{x})_i = (A\mathbf{x})_i - \alpha x_i$. This means that the KKT conditions (7) are violated, and this contradicts the hypothesis that $\mathbf{x}$ is a solution of (6). □

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). 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., [7]), 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). 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 1, if

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

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

5. 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) 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 $A + \alpha(\mathbf{e}\mathbf{e}' - I)$, 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.

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

Figure 1. Pseudo-code for our hierarchical clustering algorithm.

A high-level description of our algorithm is shown in Fig. 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 (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 [3]) strict local solution of (5) 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 $\text{int}(\Delta)$.

Characterizing the complexity of our algorithm is difficult since it involves the simulation of a dynamical system. However, we have observed experimentally that it converges quickly, and this is also confirmed by earlier empirical findings (see, e.g., [3, 17]).

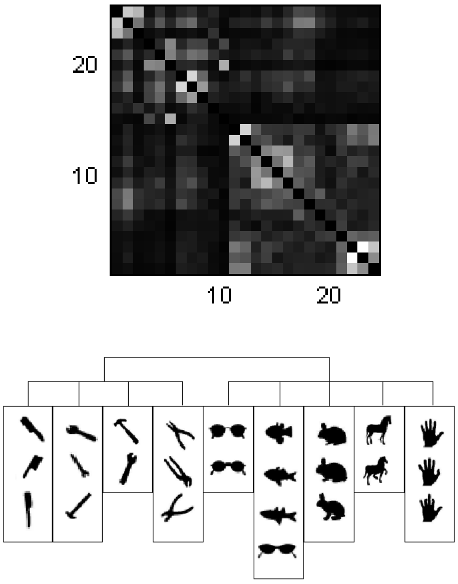

Figure 2. Top: Similarity matrix used in the experiments described in Section 6.1. Bottom: Hierarchy produced by our algorithm.

6. An example: Organizing a shape database

We illustrate the potential of our approach on the problem of organizing an image database. To this end, we used three different similarity matrices reported in the literature derived from as many binary shape databases. In all experiments, the algorithm took only a few seconds to return a hierarchy on a machine equipped with a 750 MHz Intel Pentium III.

6.1. Using Luo et al.'s similarities

The first dataset we applied our algorithm on was used by Luo et al. [11], 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 [11], shapes were abstracted in terms of rooted shock-trees [23], 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 [11, Fig. 1] and scaled properly) is shown in Fig. 2 (top). 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 shapes. 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. 2 (bottom). 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. As for 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.

These results are substantially better than those reported by Luo et al. in [11]. 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. In addition, their algorithm does not offer any hierarchical interpretation.

6.2. Using Klein et al.'s similarities

A second series of experiments was conducted on the database used by Klein et al. in [9], 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. 3 (top) shows the similarity matrix obtained by transforming the distances presented in [9, Table 1] into affinities using a customary exponential transformation. The order of the entries is the same as the one given above.

Fig. 3 (bottom) 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. 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.

6.3. 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 [4, 5, 6] which

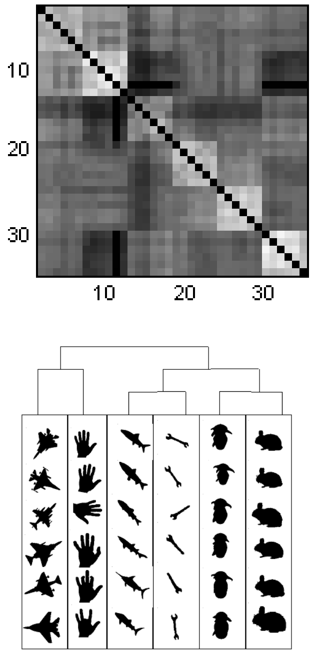

Figure 3. Top: Similarity matrix used in the experiments described in Section 6.2. Bottom: Hierarchy produced by our algorithm.

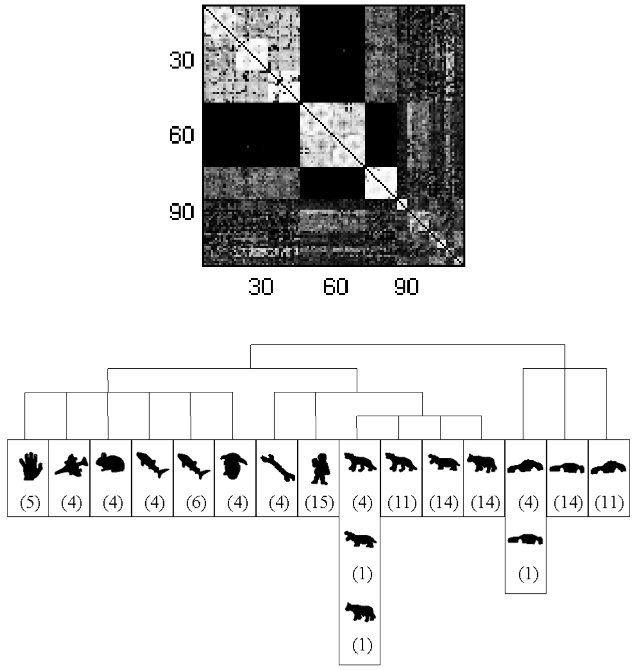

Figure 4. Top: Similarity matrix used in the experiments described in Section 6.3. Bottom: Hierarchy produced by our algorithm.

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 [5], 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. The resulting matrix is shown in Fig. 4 (top), with the entries ordered as above.

The results of applying our algorithm on this database are shown in Fig. 4 (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 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 [18] (see [4, p. 91]), which does not suggest a natural hierarchical interpretation of the data, and are comparable with the ones produced by typical/normalized cut algorithms [6, 22] (see [4, Sect. 4.4.] for details), 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 [4, 6], 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, the complete-link approach fails completely on these data [4, p. 91]. 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.

7. Conclusions

We have developed a new framework for pairwise hierarchical clustering centered around the notion of a dominant set, a newly introduced graph-theoretic concept that has proven to be relevant in partitional clustering and image segmentation problems. We have considered a family of parameterized (continuous) quadratic programs 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. These properties have motivated our hierarchical clustering algorithm, which is based on the idea of varying the regularization parameter in a principled way during the clustering process. Straightforward dynamics from evolutionary game theory are used to locate the solutions of the quadratic programs at each level of the hierarchy.

The approach is general and can be applied to a variety of computer vision and pattern recognition problems. We have demonstrated its potential for the problem of image database organization. Experiments with three different similarity matrices (and databases) reported in the literature have been conducted, and the results have shown the effectiveness of our approach.

Acknowledgements. We would like to thank Y. Gdalyahu for providing us the data used in the experiments reported in Section 6.3.

References

[1] S. Aksoy and R. M. Haralick. Graph-theoretic clustering for image grouping and retrieval. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 1999.

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

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

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

[5] 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.

[6] Y. Gdalyahu, D. Weinshall, and M. Werman. Self-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.

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

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

[9] 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.

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

[11] 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*, volume 1, pages 912–919, 2001.

[12] D. Miller and S. W. Zucker. Efficient simplex-like methods for equilibria of nonsymmetric analog networks. *Neural Computation*, 4(2):167–190, 1992.

[13] 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.

[14] 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.

[15] M. Pavan and M. Pelillo. Unsupervised texture segmentation by dominant sets and game dynamics. In *Proc. IEEE Int. Conf. on Image Analysis and Processing*, 2003 (in press).

[16] M. Pavan, M. Pelillo, and E. Jabara. On the combinatorics of standard quadratic optimization. Forthcoming.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

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

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

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

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

[25] 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.

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