A Hierarchical Clustering Algorithm Based on the Hungarian Method

Jacob Goldberger
School of Engineering
Bar-Ilan University, Israel

Tamir Tassa
Division of Computer Science
The Open University, Israel

Abstract

We propose a novel hierarchical clustering algorithm for data-sets in which only pairwise distances between the points are provided. The classical Hungarian method is an efficient algorithm for solving the problem of minimal-weight cycle cover. We utilize the Hungarian method as the basic building block of our clustering algorithm. The disjoint cycles, produced by the Hungarian method, are viewed as a partition of the data-set. The clustering algorithm is formed by hierarchical merging. The proposed algorithm can handle data that is arranged in non-convex sets. The number of the clusters is automatically found as part of the clustering process. We report an improved performance of our algorithm in a variety of examples and compare it to the spectral clustering algorithm.

1 Introduction

Automatic grouping of objects into clusters is one of the fundamental problems in machine learning and in other fields of study. In many approaches, the first step toward clustering the data-set is extracting a feature vector from each object. The clustering is then performed by applying various clustering algorithms on these feature vectors. Two commonly used methods are K-means and applying the EM algorithm to learn a Mixture of Gaussians density (see e.g. [1]). Although these iterative methods may suffer from the problem of local optima, they provide high quality results when the data is organized in convex sets (blobs). This situation is characterized by the fact that the members of the same cluster are all similar to a single point which is the cluster centroid and they are all far apart from centers of other clusters. When the data is arranged in non-convex sets (e.g. concentric circles) these algorithms tend to fail. In such cases, two points are in the same cluster, even tough they are far apart, if there is a path of locally similar points that connects them. To find clusters in such cases, other approaches, essentially global, are required. Another problem with the above mentioned methods is that they require explicit feature-vector representation of the objects. However, sometimes such an information is not available and, instead, only pairwise similarity information is provided.

An alternative clustering approach that attracted much attention in recent years is spectral clustering [10, 7, 13]. The spectral clustering algorithm can handle non-convex arrangements of clusters and it is based on pairwise similarity information. It first obtains a low-dimensional representation of the data and then uses the K-means algorithm to carry out the clustering. More explicitly, spectral clustering proceeds as follows. First, it translates the pairwise distance information, $d_{i,j}$, into an affinity matrix W whose entries are given by

$$W_{i,j} = \begin{cases} e^{-d_{i,j}/2\sigma^2} & i \neq j \, , \\ 0 & i = j \, ; \end{cases}$$

here, σ is a scaling factor. Letting D be the diagonal matrix whose ith diagonal element is the sum of W's ith row, we define $L = D^{-1/2} W D^{-1/2}$. The spectral clustering algorithms needs to receive as an input the number of clusters. Let k be that number and let $\mathbf{x}_1, \ldots, \mathbf{x}_k \in \mathbb{R}^n$ be the k largest eigenvectors of L. Letting $X = [\mathbf{x}_1, \ldots, \mathbf{x}_k]$, we define Y to be the $n \times k$ matrix that is obtained from X by normalizing each of its rows to have an ℓ_2-unit length. The rows of Y define n points on the ℓ_2-unit-sphere in $\mathbb{R}^k$. The spectral clustering algorithm then proceeds to cluster those n points into k clusters using K-means. Finally, the resulting clustering of those n points in $\mathbb{R}^k$ is projected onto the n points in the input data-set.

Apart from spectral clustering, other pairwise clustering algorithms were successfully applied. E.g., the method presented in [12] is based on the loopy Belief-Propagation algorithm that is applied on the affinity matrix. As another example, the study in [5] presents a clustering technique that applies deterministic annealing methods combined with the EM algorithm. A general description on existing clustering approaches can be found in [1, 14].

In this paper we introduce a simple novel pairwise clustering algorithm. We form clusters by explicitly constructing a short closed path (cycle) through all points in the same cluster, such that adjacent nodes in the path correspond to similar points in the data-set. (A similar intuitive definition of similarity was proposed by Fischer and Buhmann [4] who defined a pairwise distance based on the shortest path between the two points.) To find such paths we utilize the well known Hungarian method that was originally proposed as an algorithm for the assignment problem [6]. In the proposed algorithm we iteratively apply the Hungarian method on the graph that is defined by the pairwise distance matrix. This clustering algorithm, which we dub *the Hungarian clustering algorithm*, produces a hierarchical clustering of the input data-set, as opposed to the spectral clustering algorithm that does not provide such an hierarchical structure. Another advantage of our algorithm is that it decides automatically the number of clusters. Clustering results are similar or better than those obtained by using the spectral clustering algorithm.

The paper proceeds as follows. In Section 2 we provide the necessary theoretical background; we describe there the assignment problem and the closely related cycle cover problem that are solved by the Hungarian method. In Section 3 we suggest a pairwise clustering algorithm that is based on the Hungarian method. Experimental results on simulation and UCI repository data are presented in Section 4.

2 Theoretical Foundations

In this section we present graph theory definitions and theorems, related to the Hungarian method, that form the theoretical foundation needed for the main body of the paper.

2.1 The Assignment Problem and the Cycle Cover Problem

Let A be a square $n \times n$ matrix. A transversal of A is a selection of n entries, one in each row and each column. Let S_n be the group of all $n!$ permutations on n elements. The assignment problem seeks for a minimal transversal; i.e., a permutation $\pi \in S_n$, such that $\sum_{i=1}^{n} A_{i,\pi(i)}$ is minimal. Viewing A as the matrix of weights associated with a complete bipartite graph with n vertices in each of its parts, such a permutation represents a minimal-weight perfect matching in the graph. This minimization problem can be expressed as a linear programming problem:

$$\begin{aligned} \min_{X} \quad & \mathrm{Tr}(X^{\top} A) \\ \text{s.t.} \quad & \sum_{i} X_{ij} = 1 \quad \forall j \quad , \quad \sum_{j} X_{ij} = 1 \quad \forall i \quad , \quad X_{ij} \geq 0 \quad \forall i, j \end{aligned}$$

where the minimization is performed over all the $n \times n$ double-stochastic matrices X. The fact that the above linear programming problem has an integral optimal solution (namely, X is a permutation matrix) follows from a classical theorem [2]. The Kuhn-Munkers algorithm, a.k.a The Hungarian method [6, 8], is an algorithm that solves this problem in time $O(n^3)$. A description of the algorithm appears in Section 2.2.

We now turn our attention to a different problem. Let $G = (V, E)$ be a (possibly directed) graph with weights on the edges, $w : E \to \mathbb{R}$. The cycle cover problem seeks a minimal-weight subset of edges, $E' \subset E$, that constitutes a union of cycles, where each vertex belongs to exactly one cycle. We show next that this problem is equivalent to the assignment problem. Given an instance of the cycle cover problem, $\langle G, w \rangle$, the corresponding instance of the assignment problem is the square matrix A of dimension $n = |V|$, where $A_{i,j} = w((V_i, V_j))$ if $(V_i, V_j) \in E$ and $A_{i,j} = \infty$ otherwise, where ∞ denotes hereinafter a very large number (note that if G is an undirected graph then A is symmetric). If $\pi \in S_n$ is an optimal permutation that describes a minimal transversal in the matrix A, it induces the following minimal-weight cycle cover in G: $E' = \{(V_i, V_{\pi(i)}) : 1 \leq i \leq n\}$. Vice versa, given an instance of the assignment problem, $\langle n, A \rangle$, the corresponding instance for the cycle cover problem is the graph $G = (V, E)$ where $V = \{1, \ldots, n\}$, $E = V \times V$, and $w(i, j) = A_{i,j}$. If E' is a minimal-weight cycle cover, then the in-degree and out-degree of each vertex in $G' = (V, E')$ are both one. Hence, there exists a permutation $\pi \in S_n$ such that $E' = \{(V_i, V_{\pi(i)}) : 1 \leq i \leq n\}$. This permutation describes a minimal transversal in the matrix A.

There is a close relation between the cycle cover problem and the travelling salesman problem (TSP). In fact, the TSP is the cycle cover problem with the additional constraint that the cover must include a single cycle. Since the TSP problem is NP-hard, the Hungarian method is sometimes used in order to solve it in an approximate manner. First, it is applied in order to find a minimal-weight cycle cover; then, the distinct cycles in that cover are connected to form a single cycle, the length of which approximates that of the shortest-length cycle. The fact that the Hungarian method finds a minimal-weight cycle cover, where the cover may include any number of distinct cycles, is a drawback when trying to solve the TSP problem; however, it is exactly this feature that renders the Hungarian method suitable as a tool for clustering, since the found cycles indicate the underlying clustered structure of the data-set. In the context of clustering, TSP was previously used by Climer and Zhang [3] who applied (single-cycle approximations of) the TSP for rearrangement clustering which is the problem of arranging a set of objects in a linear order such that the differences between adjacent objects is minimized.

2.2 The Hungarian Method

Here, we review the Hungarian method [6, 8], which is the basic building block of our clustering algorithm. Let A be a $n \times n$ matrix. The following algorithm finds a permutation $\pi \in S_n$ that minimizes the expression $\sum_i A_{i,\pi(i)}$. In this algorithm, the entries of the matrix A are being modified repeatedly. Zero entries in the modified matrix may be either marked, by a star or by a prime, or unmarked. In addition, each row or column in the matrix may be either covered or uncovered. Initially, there are no starred or primed entries in the matrix and none of the rows or columns is covered.

1. For each row in the matrix A find its minimal entry and subtract it from all entries in that row.
2. For all $1 \leq i, j \leq n$, if $A_{i,j} = 0$ then star that zero entry, unless there is already a starred zero in the same row or in the same column.
3. Cover each column that contains a starred zero. If all columns are covered, go to Step 7.
4. Repeat the following procedure until there are no uncovered zeros left and then go to Step 6: Find an uncovered zero and prime it. If there are no starred zeros in the same row as this primed zero, go to Step 5. Otherwise, cover this row and uncover the column containing the starred zero.
5. Construct a series of alternating primed and starred zeros as follows. Let z_0 be the uncovered primed zero that was found in Step 4. Let z_1 be the starred zero in the column of z_1 (if any). Let z_2 be the primed zero in the row of z_1 (there will always be one). Continue to construct this series of alternating primed and starred zeros until it terminates with a primed zero that has no starred zero in its column. Unstar each starred zero of the series, star each primed zero of the series, erase all primes and uncover all rows and columns in the matrix. Go to Step 3.
6. Find the smallest uncovered value, add it to every entry in each covered row, and subtract it from every entry in each uncovered column. Go to Step 4.

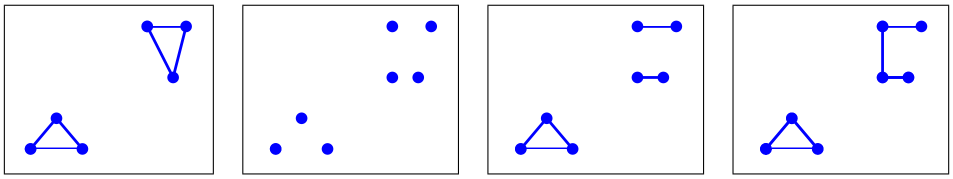
7. At this stage, in each row of the matrix, as well as in each column, there is exactly one starred zero. The positions of the starred zeros describe an optimal permutation $\pi \in S_n$. Output this permutation and stop.

3 The Hierarchical Clustering Algorithm

We now turn to describe a novel and simple algorithm for clustering that uses the Hungarian method as its basic primitive. Let (V, E) be a complete graph of n vertices, and let $d : E \to \mathbb{R}$ be a distance function. Specifically, we let $d_{i,j}$, $i \neq j$, denote the distance between the ith and jth vertices; the diagonal distances are set to $d_{i,i} = \infty$, $1 \leq i \leq |V|$, to indicate the non-existence of loops in the graph (i.e., edges that connect a vertex to itself). A typical example is that in which V consists of points in a metric space and d is the metric.

The algorithm proposed herein applies the Hungarian method to solve the minimal-weight cycle cover problem for this graph. The idea is that a minimal-weight cycle cover will be composed of cycles that connect only close points, namely, that such a cover will identify the underlying clusters of which V is composed. For example, if V is the set of six points in the plane depicted in Figure 1a, the optimal cycle cover, shown in Figure 1a, indeed identifies the two clusters.

However, simply applying the Hungarian method once on the original set of points V might result in a too large number of clusters. For example, the optimal cycle cover for the seven points in Figure 1b is depicted in Figure 1c. Here, rather than connecting the four points in the north-eastern corner through a single cycle, they are connected via two separate cycles. This cover wrongly identifies three clusters where there are perceptually only two clusters.



(a) (b) (c) (d)

Figure 1: (a) Applying the Hungarian clustering algorithm to a six-point set reveals the two clusters. (b) A seven-point set consisting of two clusters. (c) Clustering results after the first iteration of the Hungarian clustering algorithm. (d) Final clustering results.

The phenomenon demonstrated in Figure 1c is very common and, in fact, it is inevitable for clusters consisting of an even number of points. We proceed to show that if a cluster consists of an even number of points, then a closed cycle connecting all those points is never the minimal-weight cycle cover for that cluster. Namely, an optimal cycle cover for data-sets having clusters of even size, will always cover such even-size clusters with multiple cycles, rather than with a single cycle. Indeed, assume a cluster that consists of $2n$ points and let $1 \to 2 \to 3 \to \cdots \to 2n \to 1$ be an optimal single cycle cover for that cluster (namely, that cycle is a solution of the corresponding travelling salesman problem). Letting $d_{i,j}$ denote the distance between the ith and jth points (where $j = 2n+1$ is identified with $j = 1$), it is easily verified that:

$$\sum_{i=1}^{2n} d_{i,i+1} \quad = \quad \sum_{i \text{ even}} d_{i,i+1} + \sum_{i \text{ odd}} d_{i,i+1} \geq 2 \min \left(\sum_{i \text{ even}} d_{i,i+1} \, , \sum_{i \text{ odd}} d_{i,i+1} \right) \, .$$

Hence, one of the two pairwise cycle covers $\{1 \leftrightarrow 2, 3 \leftrightarrow 4, \cdots, 2n-1 \leftrightarrow 2n\}$ or $\{2 \leftrightarrow 3, 4 \leftrightarrow 5, \cdots, 2n \leftrightarrow 1\}$ is better than the single cycle cover (even though it is not necessarily the minimal-weight cycle cover).

In view of the above, we apply the Hungarian method in an iterative manner. Denote the original complete graph by $G_0 = (V_0, E_0)$. After applying the Hungarian method to that graph, we consider a new complete graph $G_1 = (V_1, E_1)$, where V_1 consists of the cycles in the optimal cover that was found for G_0 and E_1 is the set of all $\binom{|V_1|}{2}$ edges between those cycles. We then evaluate the corresponding distance function, namely, the distance between the cycles, and apply the Hungarian method once again, this time to the new reduced graph G_1. We proceed in this manner to construct a sequence of complete graphs, $G_i = (V_i, E_i)$, where V_i stands for the set of cycles in an optimal cycle cover for the graph G_{i-1}. Hence, each vertex in V_i is nothing but a cluster of points in the original set of points V.

As the graphs G_i do not include loops, cycles of size one are prohibited. Hence, the optimal cycle cover issued by the Hungarian method will connect vertices into cycles of size two at the least. This implies that $|V_i| \leq |V_{i-1}|/2$. In addition, since each vertex in V_i, viewed as a cluster of V-points, is a union of some vertices (clusters) in V_{i-1}, the sequence $\{V_i\}$ is a monotone sequence of clusterings for V, where each clustering in the sequence is coarser than the previous one. Therefore, this hierarchical clustering process will eventually reach a stage s where $|V_s| = 1$, namely, where all points in V are clustered into one cluster.

In order to avoid that outcome, where all clusters collapse into a single trivial cluster, we need to examine in every stage of this process each of the clusters in V_i and identify clusters that are already perceptually complete. Such clusters should be removed from subsequent clustering steps so that they are not forced to be merged into coarser clusters. Therefore, we need to distinguish complete clusters from those that are still incomplete. We describe below a test that we apply in order to identify such complete clusters. A cluster that has been identified as complete will be removed from the clustering process in the subsequent stages and will be output as one of the final clusters of the original graph V.

In order to identify complete clusters, we carefully compute the distance between clusters. Our basic definition of distance between clusters follows the usual topological definition of distance between sets, namely, the minimal distance between points in the two sets. However, if two clusters are deemed ”too far”, we define their distance as infinite, so that they will not be connected through a cycle in subsequent applications of the Hungarian method. Then, after computing the full distance matrix between clusters, if some cluster is identified as ”too far” from all other clusters, we consider it as a complete cluster and remove it from subsequent clustering procedures by forcing the Hungarian method to return an optimal cycle cover with that cluster as a singleton cycle in the cover. The ability to declare clusters as complete is the measure against total collapse of the entire graph into one cluster. With this safety measure, we continue our iterated Hungarian clustering until all clusters are declared complete (i.e., until we reach a stage s where $V_s = V_{s-1}$).

Going back to Figures 1b-1d, they depict the two iterations in the hierarchical Hungarian clustering for the toy data-set $G_0 = (V_0, E_0)$, given in Figure 1b, where $|V_0| = 7$. Applying the Hungarian method to G_0 we arrive at the intermediate clustering depicted in Figure 1c, $G_1 = (V_1, E_1)$, where $|V_1| = 3$. Applying the Hungarian method once again to G_1, the cluster of three points in the south-western corner is declared complete, while the two clusters in the north-eastern corner are connected through a cycle. This brings us to the final clustering depicted in Figure 1d, $G_2 = (V_2, E_2)$, where $|V_2| = 2$.

Next, we describe the crucial ingredient of distance computation. Assume that in a given round of this algorithm, the clusters are $C_i = \{c_{i,1}, \ldots, c_{i,\ell_i}\}$, $1 \leq i \leq k$ (namely, the ith cluster includes ℓ_i points from V and $V = \dot\bigcup_{i=1}^{k} C_i$.). Then the distance matrix $d(C_i, C_j)_{1 \leq i,j \leq k}$ is defined as follows:

• Computing $d(C_i, C_j)$ for $1 \leq i \neq j \leq k$: Set $d_{i,j} = \min\{d(c_{i,r}, c_{j,s}) : 1 \leq r \leq \ell_i, 1 \leq s \leq \ell_j\}$. Let $c_{i,r_0} \in C_i$ and $c_{j,s_0} \in C_j$ be two points such that $d_{i,j} = d(c_{i,r_0}, c_{j,s_0})$. Let T be some fixed integral parameter. Then if C_i includes at least T points whose distance to c_{i,r_0} is smaller than $d_{i,j}$, or if C_j includes at least T points whose distance to c_{j,s_0} is smaller than $d_{i,j}$, set $d(C_i, C_j) = \infty$; otherwise, $d(C_i, C_j) = d_{i,j}$.

• Computing $d(C_i, C_i)$ for $1 \leq i \leq k$: If C_i has an infinite distance to all other clusters, set $d(C_i, C_i) = -\infty$; otherwise, $d(C_i, C_i) = \infty$.

THE HUNGARIAN CLUSTERING ALGORITHM

Input: a matrix $n \times n$ of pairwise distances

Algorithm:

1. Set the number of clusters to be $k = n$ and $C_j = \{j\}$, $1 \leq j \leq k$.
2. Compute the distance matrix $d(C_i, C_j)$ for all $1 \leq i, j \leq k$.
3. Apply the Hungarian method to find the optimal cycle cover in the complete weighted graph with vertices $\{C_1, \ldots, C_k\}$.
4. Update the number of clusters, k, to equal the number of cycles in the optimal cycle cover that was found above. For all $1 \leq i \leq k$, the new ith cluster is the union of the old clusters that were connected through the ith cycle in the cover.
5. If the new number of clusters is less than the previous number of clusters, go to Step 2 for another iteration. Otherwise (namely, the last cycle cover is composed of singleton cycles), stop and output the clustering $\{C_1, \ldots, C_k\}$.

Our algorithm is parametric, as it depends on the parameter T that has to be manually tuned. That parameter is used for the identification of clusters that are too far apart. Two clusters, C_i and C_j, are considered "too far" (whence $d(C_i, C_j)$ is set to ∞) if the minimal distance between points in those two clusters is too large in comparison with the internal distances in at least one of these clusters. Regarding the distances along the diagonal, we usually have $d(C_i, C_i) = \infty$, in order to indicate to the Hungarian method that the graph has no loop in C_i (i.e., an edge that connects the vertex C_i to itself). However, if at some stage a cluster C_i is found to be too far from all other clusters, i.e. $d(C_i, C_j) = \infty$ for all $j \neq i$, we set $d(C_i, C_i) = -\infty$ so that any optimal cycle cover will include the singleton cycle $\{C_i\}$. Note that once a cluster has been found complete, it will remain complete in all subsequent rounds.

Our parameter T plays a similar role to that of the scaling parameter σ of the spectral clustering algorithm due to Ng, Jordan and Weiss [10]. Both parameters, in both methods, have to be manually tuned in order to achieve best results. In [15], Zelnik-Manor and Perona proposed to use local scaling parameters σ_i, $1 \leq i \leq n$, one for each input point (instead of a single global parameter σ) and then introduced a way in which those parameters may be automatically tuned. However, the automatic tuning depends on yet another parameter K that has to be manually tuned. (In fact, the definition of our parameter T is quite similar to that of the integral parameter K of Zelnik-Manor and Perona.) In all of the experiments in [15] they chose the value $K = 7$, but it may have to be tuned differently in clustering problems of different sizes or characteristics. Hence, it is quite hard to completely escape from manually tuning parameters in clustering algorithms. To the best of our knowledge, all clustering algorithms rely upon some manually tuned parameters. Such parameters may be tuned, for example, by experimentation on cross-validation data. Having said that, we found empirically that our algorithm is not highly sensitive to the exact value of T. In all of our experiments we set $T = 7$ and it worked well in both simulation and real data.

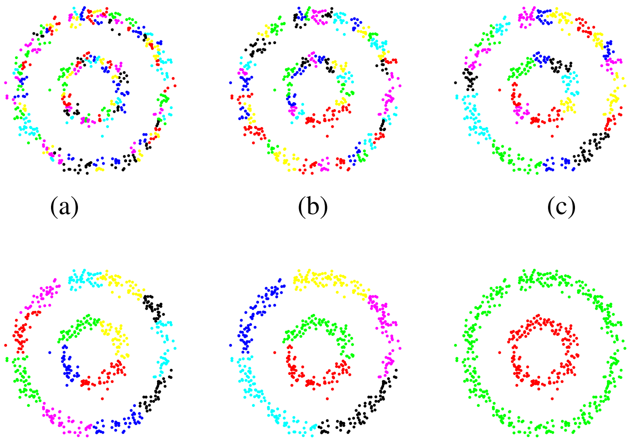
Apart from σ, the spectral clustering algorithm depends on yet another input parameter which is the actual number of clusters in the given data-set. The proposed Hungarian clustering algorithm, on the other hand, does not depend on that additional input parameter. This is a significant advantage of the Hungarian clustering algorithm since the number of clusters is not always available.

Another advantage of our algorithm with respect to the spectral algorithm is that it provides *hierarchical* clustering. This advantage was exploited by Jaffe et. al. in [9]. That study proposes a method for generating summaries and visualization for large collections of geo-referenced photographs. Since that method depends on a clustering of the data-set of photograph locations and, furthermore, it requires the hierarchical structure of each cluster, it utilizes the Hungarian clustering algorithm.

4 Experimental Results

4.1 Results on Simulation Data

We start with a detailed illustrative example of the clustering algorithm. Figure 2 demonstrates the iterative progress of the Hungarian clustering algorithm when applied to a data-set consisting of two concentric circles. The distance used is the Euclidean



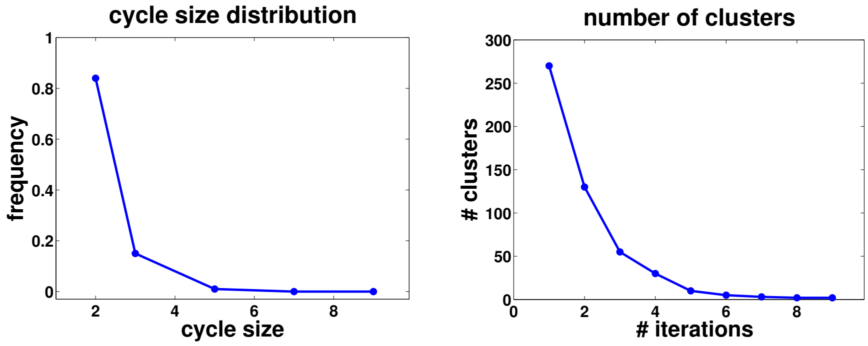
(a) (b) (c)

(d) (e) (f)

Figure 2: Intermediate clustering results obtained in the six iterations of the Hungarian clustering algorithm for a data-set of 600 points arranged in two concentric circles.

distance between the points. The Hungarian clustering algorithm ran in this case for six iterations until it issued its final clustering. The number of points in the data-set was 600. Figures 2a through 2f depict the intermediate clusterings at the end of each of the six iterations for this data-set (note that we used a pallet of only 8 colors in these drawings, whence there are different clusters that are indicated by the same color). From Figure 2 we can see that the algorithm is usually very local and hierarchical. Namely, the cycles in the optimal cycle cover issued by the algorithm are typically very small and most of the time they are of minimal length, i.e. 2. Eventually, in the last iteration, the algorithm issues global merging of the clusters.

Figure 3 illustrates the essential local behavior of the algorithm in a statistically systematic manner. It presents statistics of cluster sizes and number of clusters as obtained during the execution of the iterative Hungarian clustering algorithm. The statistics represents 100 applications of the algorithm on random data-sets consisting of 600 points that are arranged in two concentric circles (Figure 2 shows one of those data-sets). Very similar statistical results were observed for data-sets with other cluster patterns. Figure 3a presents the distribution of the cycle sizes in the optimal cycle covers issued by the Hungarian method. As can be seen from Figure 3a, more than 80% of the cycles in the optimal cycle covers are of length two. Namely, a typical step of the iterative clustering algorithm is merging two adjacent clusters. This shows that most of the time the algorithm is operating in a local and hierarchical manner. The same behavior can be observed by analyzing the average number of clusters in each iteration. Figure 3b presents the relatively slow drop in the average number of clusters as a function of the iteration number (in both graphs, the error bars are too small to be observed).



(a) (b)

Figure 3: Statistics of the iterative clustering algorithm. (a) The histogram distribution of the cycle sizes that are obtained in the iterative Hungarian clustering algorithm. (b) The number of clusters as a function of the iteration.

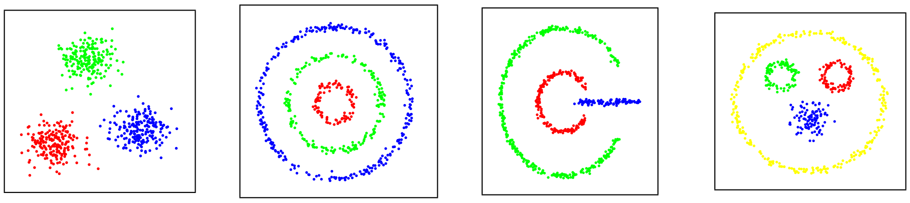
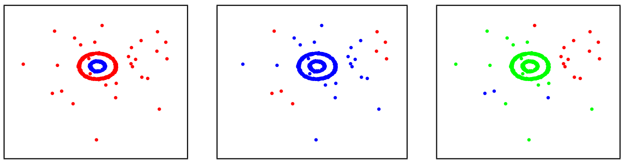


Figure 4: Examples of clustering results of the Hungarian clustering algorithm. The NJW algorithm [10] obtained the same results.

Next, we show several clustering examples in which both the Hungarian and the spectral clustering algorithms issued identical clusterings that coincide with human perception. We have used the variant of the spectral clustering approach proposed by Ng, Jordan and Weiss (NJW) [10]. Figure 4 depicts four such examples. Note that in our algorithm the number of clusters was automatically inferred from the data and it was not required as an external input. Although in most cases that we checked, our algorithm and the NJW algorithm issued identical clusterings, in some other cases their performance differed. Figure 5 presents such an example where there are two concentric circles with additional outlier points. Our algorithm overcame the noise. The NJW algorithm, on the other hand, failed despite the fact that it was given the number of clusters and the variance of the Gaussian kernel was manually optimized. Spectral clustering can be still successfully applied by using the self-tuning method [15].



(a) (b) (c)

Figure 5: Clustering results on noisy data (a) The Hungarian clustering algorithm. (b,c) The NJW spectral algorithm for 2 and 3 clusters, $\sigma = 0.04$.

4.2 Results on Real Data

Next, we evaluated the performance of our clustering algorithm on standard databases. In our experiments we used four data-sets from the UC Irvine repository [1]. The data-sets we used were Wine (178 points, 13 features, 3 classes), Iris (150 points, 4 features, 3 classes), Soybean (307 points, 35 features, 19 classes) and Ionosphere (351 points, 33 features, 19 classes). The distance that was used by the Hungarian clustering algorithm in all of the experiments was the Euclidean distance between the feature vectors. As it was stated before, we found empirically that our algorithm is not highly sensitive to the exact value of T. In all of our experiments we set $T=7$. The clustering performance of the Hungarian clustering algorithm is compared to that of the NJW [10] algorithm. It should be noted that while the parameter T was fixed in all runs of the Hungarian clustering algorithm, the NJW scaling parameter σ was manually optimized separately for each data-set. Another important difference is that in the NJW algorithm the number of the clusters is given as input, while in our algorithm the number of clusters is found as part of the learning procedure.

The comparison between our algorithm and the NJW algorithm was done using the Rand score [11] which is a standard measure for the clustering quality. For the sake of completeness, the variant of the Rand score that we used is as follows: Let $\mathcal{C}_1$ stand for the true clustering of the n points and $\mathcal{C}_2$ stand for the clustering in question. Then

$$\text{Rand Score} := \frac{1}{2} \cdot \left(\frac{N_{0,0}}{N_0} + \frac{N_{1,1}}{N_1} \right) ,$$

where:

- $N_{0,0}$ is the number of pairs of points that do not belong to the same cluster neither in $\mathcal{C}_1$ nor in $\mathcal{C}_2$;
- N_0 is the number of pairs of points that do not belong to the same cluster in $\mathcal{C}_1$;
- $N_{1,1}$ is the number of pairs of points that belong to the same cluster both in $\mathcal{C}_1$ and in $\mathcal{C}_2$; and
- N_1 is the number of pairs of points that belong to the same cluster in $\mathcal{C}_1$.

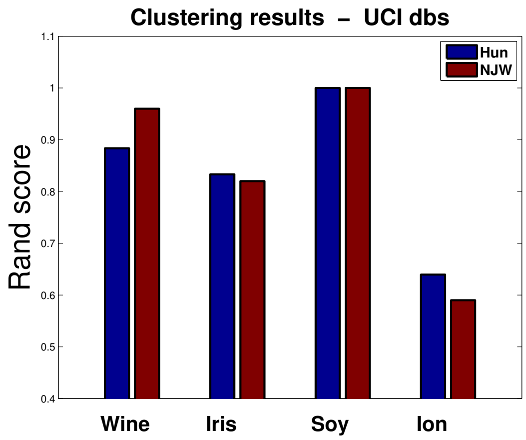


Figure 6: Unsupervised pairwise-based clustering results (using the Rand score) on UCI data-sets wine, iris, soybean, and ionosphere.

The clustering results are shown in Figure 6.

The computation time of the Hungarian clustering algorithm (using Pentium (R) 4 CPU 3.40 GHz, 1.49 GB of RAM) is less than one second for a data-set of 500 points and four seconds for a 1000-point data-set.

4.3 Robustness to Noisy Data

At the beginning of this section we claimed that our algorithm can overcome noise more easily than the NJW algorithm, and demonstrated it on few examples. The intuitive explanation is due to the local hierarchical nature of our algorithm that prevents fast propagation of clustering errors, unlike the NJW algorithm which is based on a global process of spectral decomposition of the (Laplacian of the) affinity matrix.

We proceed to establish our claim in a more systematic manner, using experiments on data-sets with simulated distance matrices. In all of those experiments we set the number of points to be $n = 500$ and randomly selected the number of clusters k to be $k \in [3, 6]$; once the number of clusters k was selected, we randomly selected the number of points n_i in each cluster, $1 \leq i \leq k$, so that $\sum_{i=1}^{k} n_i = n = 500$. Then, we generated a corresponding symmetric distance matrix of the following form:

$$D = \begin{pmatrix} B_{1,1} & B_{1,2} & \cdots & B_{1,k} \\ B_{2,1} & B_{2,2} & \cdots & B_{2,k} \\ \vdots & \vdots & \vdots & \vdots \\ B_{k,1} & B_{k,2} & \cdots & B_{k,k} \end{pmatrix}$$

The entries within a diagonal block $B_{i,i}$ stand for the distances between the points of the ith cluster. The entries within the block $B_{i,j} = B^T_{j,i}$ where $i \neq j$ stand for the

[1]http://www.ics.uci.edu/˜ mlearn/MLRepositiry.html

distances between the points of the ith and the jth clusters. The entries in the blocks along the diagonal were set to small values, while those outside of the diagonal blocks were set to large values. More specifically, the intra-cluster distances (namely, the entries in all of the diagonal blocks) were drawn from the positive normal distribution $\lambda \cdot |N(0,1)|$, for some parameter $\lambda > 0$, where $N(0,1)$ is the zero-mean, unit-variance normal distribution. On the other hand, the inter-cluster distances (namely, the entries outside the diagonal blocks) were drawn from the shifted positive normal distribution $1 + \lambda \cdot |N(0,1)|$. The parameter λ represents the amount of noise: larger values of λ represent noisier data where the clusters are more mixed and therefore less distinguishable.

We tested the two clustering algorithms for several values of λ. For each value of λ we generated 100 random matrices that comply with the selected value of λ. Figure 7 shows the average Rand scores (and error bars) for the Hungarian and the NJW clustering algorithms. As can be seen, the Hungarian clustering algorithm identifies the underlying clusters better than the NJW algorithm. While the latter algorithm collapses for values of $\lambda \geq 5$, the decrease in the score of the Hungarian clustering algorithm is much milder and even for very noisy data ($\lambda = 10$) it still produces meaningful results.

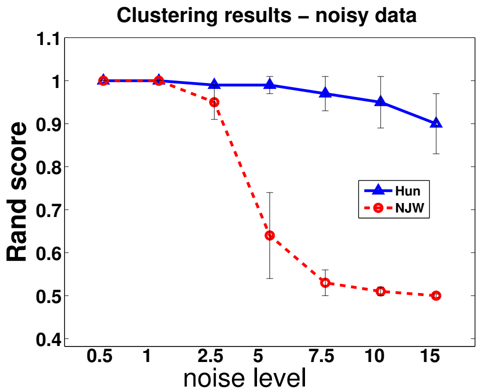


Figure 7: Clustering accuracy tested on block pairwise-distance matrices with different levels of noise.

5 Discussion

In this study we presented a novel approach for unsupervised clustering that is based on the Hungarian method. The algorithm produces a hierarchical clustering and the number of clusters can be decided automatically. We assume nothing about the structure of the connected clusters and the decision boundaries between them. Clustering results are similar or better than those obtained by using the spectral clustering algorithm. Our algorithm, however, uses only elementary arithmetic operations (integer additions, subtractions, and comparisons), as opposed to the spectral clustering algorithm that performs spectral analysis of real matrices. There is no need to provide the true number of clusters to the proposed algorithm. It relies just on a single discrete parameter T that can be easily determined in real cases by cross validation.

The polynomial computational complexity of the Hungarian method can be further reduced by using a relaxed version of that method in order to obtain an approximate optimal cycle cover. As demonstrated in Figure 3a, the sizes of cycles in an optimal cover is typically very small. Hence, by restricting ourselves in the distance matrix only to a small constant number of closest neighbors in every row, we may reduce the computational complexity from $O(n^3)$ to $O(n^2)$ without incurring a significant effect on the weight of the resulting cover. Ongoing work is focused on investigating the performance of that relaxed version of the Hungarian clustering algorithm. We are also investigating methods for self-tuning the parameter T that controls the number of clusters in the obtained clustering.

References

[1] C. Bishop. Pattern Recognition and Machine Learning. Springer, 2006.

[2] G. Birkhoff. Tres observaciones sobre el algebra lineal. *Revista Facultad de Ciencias Exactas, Puras y Aplicadas Universidad Nacional de Tucuman, Serie A (Matematicas y Fisica Teorica)*, 5:147–151, 1946.

[3] S. Climer and W. Zhang. Take a walk and cluster genes: A tsp-based approach to optimal rearrangement clustering. *International Conference on Machine Learning*, 2004.

[4] B. Fischer and J. M. Buhmann. Path-based clustering for grouping of smooth curves and texture segmentation. *IEEE Trans. on pattern Anal. and Machine Int.*, 2003.

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

[6] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.

[7] M. Meilă and J. Shi. Learning segmentation by random walks. *Advances in Neural Information Processing Systems*, 2001.

[8] J. Munkers. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5:32–38, 1957.

[9] A. Jaffe, M. Naaman, T. Tassa, and M. Davis. Generating summaries and visualization for large collections of geo-referenced photographs. *ACM Multimedia Workshop On Multimedia Information Retrieval*, pages 89–98, 2006.

[10] A. Ng, M. Jordan, and Y. Weiss. On specral clustering: Analysis and algorithm. *Advances in Neural Information Processing Systems*, 2002.

[11] W. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, Vol. 66, 846–850, 1971.

[12] N. Shental, A. Zomet, T. Hertz, and Y. Weiss. Pairwise clustering and graphical models. *Advances in Neural Information Processing Systems*, 2003.

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

[14] R. Xu and D. Wunsch, II. Survey of clustering algorithms. *IEEE Trans. Neural Netw.*, pp. 645-678, 2005.

[15] L. Zelnik-Manor and P. Perona. Self tuning spectral clustering. *Advances in Neural Information Processing Systems*, 2004.