Central Clustering of Attributed Graphs: Theoretical Advances in Data Clustering (Guest Editors: Nina Mishra and Rajeev Motwani)

Article *in* Machine Learning · July 2004

DOI: 10.1023/B:MACH.0000033119.52532.ce · Source: dx.doi.org

Brijnesh Jain
Technische Universität Berlin
109 PUBLICATIONS **670** CITATIONS

Machine Learning, 56, 169–207, 2004

Central Clustering of Attributed Graphs

BRIJNESH J. JAIN bjj@cs.tu-berlin.de
FRITZ WYSOTZKI wysotzki@cs.tu-berlin.de
Department of Computer Science, Technical University of Berlin, Franklinstr. 28/29, D-10587 Berlin, Germany

Editors: Nina Mishra and Rajeev Motwani

Abstract. Partitioning a data set of attributed graphs into clusters arises in different application areas of structural pattern recognition and computer vision. Despite its importance, graph clustering is currently an underdeveloped research area in machine learning due to the lack of theoretical analysis and the high computational cost of measuring structural proximities. To address the first issue, we introduce the concept of metric graph spaces that enables central (or center-based) clustering algorithms to be applied to the domain of attributed graphs. The key idea is to embed attributed graphs into Euclidean space without loss of structural information. In addressing the second issue of computational complexity, we propose a neural network solution of the K-means algorithm for structures (`KMS`). As a distinguishing feature to improve the computational time, the proposed algorithm classifies the data graphs according to the principle of *elimination of competition* where the input graph is assigned to the winning model of the competition. In experiments we investigate the behavior and performance of the neural `KMS` algorithm.

Keywords: attributed graphs, graph matching, K-means clustering, weighted maximum clique, Hopfield networks, winner-takes-all networks

1. Introduction

In most areas of pattern recognition it is common practice to represent real world objects in terms of static data structures as fixed dimensional feature vectors in an Euclidean vector space. This kind of representation is convenient because the Euclidean space offers powerful analytical tools for data analysis usually not available in other representations. Such a representation, however, is too limited for many relevant application areas including domains such as bioinformatics, chemistry, natural language processing, network analysis or text mining, for example. Embedding data of structured domains into a fixed-dimensional vector space either results in a loss of structural information or in high dimensional and *bulky* vectors. A more versatile and expressive tool for representing structured data are attributed graphs.

Attributed graphs are an underdeveloped research area due to (1) missing powerful analytical techniques for data analysis and (2) high computational complexity when measuring structural similarities between graphs. Referring to the former problem, it is not clear how to embed dynamic data structures like graphs into a vector space without loss of structural information. Therefore it is not straightforward to provide well defined characterizations of basic concepts like the mean or variance of a set of graphs. The latter problem of measuring the structural similarity is more generally referred to as the *graph matching problem*. It is well known that the graph matching problem is NP-complete (Garey & Johnson, 1979).

Thus, clustering large set of graphs is still widely unexplored and one of the most challenging problems in structural pattern recognition. Investigations on clustering of graphs has been recently revitalized taking into account their practical importance (Günter & Bunke, 2002a, 2002b; Hagenbuchner, Sperduti, & Tsoi, 2003; Lozano & Escolano, 2003a, 2003b; Luo et al., 2001a, 2001b; Luo, Wilson, & Hancock, 2003; Sanfeliu, Serratosa, & Alquezar, 2002). Examples of graph clustering problems are the organization of large structural databases (Sengupta & Boyer, 1995), discovering of shape categories (Luo et al., 2001a) and the view structure of objects (Luo, Wilson, & Hancock, 2003), or in the construction of nearest neighbor classifiers for predicting the mutagenicity of aromatic compounds (Schädler & Wysotzki, 1999). To illustrate the practical importance of clustering graphs in more detail, we consider a simple example from computer vision, where it is common practice to represent rotation- and translation invariant high level descriptions of images in terms of attributed graphs. At a high description level, vertices may describe elementary objects like houses, trees, or persons, and edges represent the spatial relationship of the corresponding objects. Usually noise occurs in images having the same semantic content due to different light conditions and view angles. Unsupervised methods for learning the underlying class structure on images represented in terms of attributed graph is of increasing importance in order to structure datasets with respect to semantic content or solve classification tasks.

The lack of analytical methods in the domain of graphs led to the formulation of clustering problem as *pairwise clustering* (Hofmann & Buhmann, 1997). This approach partitions a set of graphs into clusters using pairwise proximity values. Since pairwise proximities can be defined in any abstract space equipped with a proximity measure, this procedure provides a bridge between standard and structural pattern recognition methods (Goldfarb et al., 1995).

Once a set of pairwise distances between graphs are supplied, different directions to extract the hidden cluster structure from a given data set have been pursued. Schädler and Wysotzki applied a hierarchical cluster algorithm for grouping a dataset of chemical compounds (Schädler & Wysotzki, 1999). Sanfeliu et al. also used hierarchical clustering of attributed graphs for 3D object recognition (Sanfeliu, Serratosa, & Alquezar, 2000, 2002). Hancock and his co-workers used an EM-style algorithm to discover shape categories (Luo et al., 2001a) and to find view structures of polyhedral objects (Luo et al., 2001b). Adopting the pairwise clustering approach, however, involves measuring the pairwise distances of the graphs. Provided that the data set consists of N samples, pairwise clustering requires the solution of $O(N^2)$ NP-complete graph matching problems which might be intractable for large data sets.

A simple way around the high computational cost of determining pairwise structural distance values is to map the graphs to vectors in Euclidean space. After embedding the graphs into Euclidean space, a whole plethora of clustering methods can be utilized. In Luo, Wilson, and Hancock (2003), transformed graphs to vectors using spectral properties. The resulting vectors were embedded in a pattern space for cluster analysis using multidimensional scaling and principal as well as independent component analysis. Hagenbuchner, Sperduti, and Tsoi formulated self-organizing feature maps for directed acyclic graphs (DAG) (Hagenbuchner, Sperduti, & Tsoi, 2003). The essential feature of their approach is to encode DAGs into vectors using complex recursive neurons. Transforming graphs to pattern vectors comes with a loss of structural information. For this reason it is not straightforward to map graphs to vectors maintaining the underlying cluster structure of the sample graphs.

Probably one major reason for limited progress in clustering structured data inheres in the elusiveness of applying conventional central clustering methods. A first step to remove this barrier was initiated by Bunke and his collaborators. Jiang, Münger, and Bunke (1999, 2001) introduced the fundamental concepts of a weighted mean of pairs of graphs and a median graph.[1] Furthermore Bunke, Münger, and Jiang (1999) proposed a genetic algorithm for searching a median graph. Günter and Bunke applied the concept of a weighted mean to cluster graphs using self-organizing feature maps (Günter & Bunke, 2002a, 2002b). Lozano and Escolano implicitly used the concept of a a weighted mean of a set of graphs in their asymmetric clustering model for attributed graphs (Lozano & Escolano, 2003a, 2003b). Another direction to utilize central clustering for graphs is by means of a weighted minimum common supergraph as proposed by Bunke et al. (2003). Nevertheless, the sample mean as well as the weighted minimum common supergraph of a set of sample graphs are not uniquely determined. In addition, central clustering algorithms applied on graphs currently require K solutions of NP-complete graph matching problems each time an input graph is presented to the clustering algorithm where K is the number of model graphs.

This paper is concerned with both adverse effects of graphical representation in the context of central clustering, the lack of a cohesive mathematical framework for data analysis and the high computational complexity of the graph matching problem. In the first part of this contribution, we introduce the theoretical framework of a *metric graph space* to extend central clustering from the domain of real valued feature vectors to the domain of attributed graphs using the K-means algorithm as our guiding example. To facilitate *K-means for structures* (KMS) we adapt the two principal constituents of that algorithm, both based on metric properties of real valued vector spaces: We provide and theoretically justify a mechanism to (1) determine the optimal reference model of a given input graph with respect to a distortion measure, and (2) adjust the models according to a learning rule. The concept of metric graph space allows us to extend other central clustering methods to the domain of graphs, like, for example, fuzzy clustering (Bezdek, 1981; Ruspini, 1969), competitive learning (Ballard, Gardner, & Srinivas, 1987; Lippman, 1987), adaptive resonance theory (Carpenter & Grossberg, 1987a, 1987b, 1988), or self-organizing feature maps (Kohonen, 1982). Moreover the concept of metric graph spaces can also be applied to extend supervised neural learning machines from the domain of feature vectors to the domain of graphs (Jain & Wysotzki, 2003).

In the second part we propose a neural network solution to the KMS algorithm. The proposed approach avoids explicit calculation of structural distortion values to assign an input graph to its optimal reference model. Classifying the data graphs to their optimal reference models follows the principle of *elimination of competition*. The competitors are the model graphs and the winning model of the competition is considered to be the optimal reference model of the current input graph. Competition among the models is implemented by an *extended winner-takes-all* (X-WTA) network. Following this approach only one instead of K solutions of NP-complete graph matching problems is required each time an input graph is presented to the neural KMS algorithm.

The paper is organized as follows: Section 2 introduces graph theoretic concepts and defines terminology used throughout this paper. In Section 3 we review the K-means algorithm for real valued feature vectors. Section 4 proposes a neural KMS algorithm. Experimental results are presented and discussed in Section 5. Finally, Section 6 summarizes this contribution and gives an outlook for further research.

2. Terminology

Basic notations

Let $\mathcal{S}$ be a set. By $\mathcal{S}^{[2]}$ we denote the set of all ordered tuples $(i, j) \in \mathcal{S}^2$ with $i \neq j$. By $\mathbb{R}$ we denote the set of real numbers and by $\mathbb{R}_+$ the subset of all non-negative real numbers. Vectors $\boldsymbol{x} \in \mathbb{R}^n$ are denoted by bold faced lower case letters and $n \times m$ real valued matrices X are denoted by upper case letters. We write $\boldsymbol{x}^t$ and X^t for the transpose of a vector $\boldsymbol{x}$ and a matrix X, respectively. The set of all $n \times m$-matrices $A = (a_{ij})$ with entries a_{ij} from a set $\mathcal{S}$ is denoted by $\mathcal{M}_{n\times m}(\mathcal{S})$.

Linear algebra

An *inner product space* over the field $\mathbb{R}$ is a vector space over $\mathbb{R}$ which is equipped with an inner product.[2]

Let $\mathcal{V}$ be an inner product space over $\mathbb{R}$ of dimension n. A well known result from linear algebra is that a matrix $A \in \mathcal{M}_{n\times n}(\mathbb{R})$ represents an inner product if and only if it is positive definite and symmetric. Let A be the matrix representing the inner product $\langle\,,\rangle$. Then we have

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \boldsymbol{x}^t A \boldsymbol{y}$$

for all $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{V}$.

A matrix $A \in \mathcal{M}_{n\times n}(\mathbb{R})$ is *orthogonal* if $A^t A = AA^t = I_n$ where $I_n \in \mathcal{M}_{n\times n}(\mathbb{R})$ denotes the identity matrix. A matrix $A \in \mathcal{M}_{n\times n}(\mathbb{R})$ is orthogonal if and only if $\langle A\boldsymbol{x}, A\boldsymbol{y} \rangle = \langle \boldsymbol{x}, \boldsymbol{y} \rangle$ for all $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{V}$. Thus an orthogonal matrix does not only preserve orthogonality but also length. An example of orthogonal matrices are permutation matrices. A *permutation matrix* is a matrix $P = (\pi_{ij}) \in \mathcal{M}_{n\times n}(\{0, 1\})$ with

$$\sum_i \pi_{ij} = \sum_j \pi_{ij} = 1.$$

Let $\mathcal{V}$ be an inner product space over $\mathbb{R}$ with inner product $\langle\, , \rangle$. Then the direct sum $\mathcal{V}^m$ is an inner product space over $\mathbb{R}$ with inner product

$$\langle(\boldsymbol{x}_1, \dots, \boldsymbol{x}_m), (\boldsymbol{y}_1, \dots, \boldsymbol{y}_m)\rangle = \langle\boldsymbol{x}_1, \boldsymbol{y}_1\rangle + \cdots + \langle\boldsymbol{x}_m, \boldsymbol{y}_m\rangle. \tag{1}$$

Note, that a vector space $\mathcal{V}^m$ can be equipped with other inner products, which are not induced by the inner product $\langle\, , \rangle$ of $\mathcal{V}$ as in (1).

Attributed graphs

Let $\mathcal{A}$ be a set of vertex and edge attributes containing a distinguished *null attribute* $\mathbf{0} \in \mathcal{A}$. Here we assume that the attribute set $\mathcal{A}$ is an inner product space over $\mathbb{R}$, for example $\mathcal{A} = \mathbb{R}^m$. The null attribute of $\mathcal{A}$ is its zero element.

An *attributed graph* is a tuple $X = (V, \mu)$ consisting of a finite set $V \neq \emptyset$ and a function $\mu : V^2 \to \mathcal{A}$. The elements of V are the *vertices* of the graph X and the pairs $(i, j) \in V^{[2]}$ with $\mu(i, j) \neq \mathbf{0}$ are its edges. The function μ is the *attribute function* of X. An *undirected attributed graph* is an attributed graph $X = (V, \mu)$ with $\mu(i, j) = \mu(j, i)$ for all pairs $(i, j) \in V^{[2]}$. A *weighted graph* is an attributed graph with attributes from $\mathcal{A} = \mathbb{R}$. A weighted graph X is a *binary graph*, if $\mu(V^2) = \{0, 1\}$.

By $\mathcal{G}_{\mathcal{A}}$ we denote the set of attributed graphs with attributes from $\mathcal{A}$. For notational convenience we sometimes drop the subscript $\mathcal{A}$ and write $\mathcal{G}$ instead of $\mathcal{G}_{\mathcal{A}}$. By $\mathcal{G}^n$ we denote the set of all attributed graphs with $m \leq n$ vertices. The vertex set of a graph X is referred to as $V(X)$, its edge set as $E(X)$, and its attribute function as μ_X.

Vertex and edge attributes are elements of the same attribute set $\mathcal{A}$, for notational convenience. If it is required to explicitly distinguish between vertex and edge attributes, the attribute set $\mathcal{A}$ can be decomposed into two disjoint subsets.

We use a *labeling* of the vertex set V in order to deal with it conveniently. A labeling of V is a bijection $l : V \to \{1, \dots, |V|\}$ which identifies each element of V with a certain number between 1 and $|V|$. The number of vertices of a graph X is its *order*, written as $|X|$.

The (*attributed*) *adjacency matrix* of an attributed graph X of order n is a matrix $A(X) = (x_{ij}) \in \mathcal{M}_{n\times n}(\mathcal{A})$ with entries $x_{ij} = \mu_X(i, j)$.

A *clique* of an attributed graph X is a subset $C \subseteq V(X)$ with $C^{[2]} \subseteq E(X)$. A clique C of X is called *maximal*, if C is not a proper subset of any other clique of X. A *maximum clique* of X is a clique with maximal number of vertices. Clearly, a maximum clique is maximal. The converse statement, however, does not hold in general.

Let X be a weighted graph with adjacency matrix $A(X) = (x_{ij})$. Then the weight $\omega(C)$ of a clique C is defined by

$$\omega(C) = \sum_{i,j\in C} x_{ij}$$

A *maximum weighted clique* is a clique with maximum weight.

Let X and Y be two attributed graphs with adjacency matrices $A(X) = (x_{ij})$ and $A(Y) = (y_{ij})$. We call the graphs X and Y *isomorphic*, in symbols $X \simeq Y$, if there exists a bijection

$$\phi : V(X) \to V(Y), \quad i \mapsto i^{\phi}$$

such that $x_{ij} = y_{i^{\phi} j^{\phi}}$ for all $i, j \in V(X)$. Such a mapping ϕ is called *isomorphism* between X and Y.

A *permutation* acting on X is a bijection $\pi : V(X) \to V(X)$ from $V(X)$ onto itself. The image graph of a permutation π acting on X is denoted by X^{π}. The set $\mathcal{S}_X$ of all permutations acting on X is called the *symmetric group* of X. A permutation π acting on X corresponds to a relabeling of X and thus yields a reordering of its adjacency matrix. In general we have $A(X) \neq A(X^{\pi})$. Let $|X| = n$. Since we assume a labeling $V(X) = \{1, \ldots, n\}$, we write $\mathcal{S}_n$ instead of $\mathcal{S}_X$.

3. Central clustering of feature vectors

Suppose that we are given a set $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ of N samples drawn from a feature space $\mathcal{F} \subseteq \mathbb{R}^d$. The basic task of *central clustering* is to find K *cluster centers* or *models* $\mathcal{Y} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_K\} \subseteq \mathcal{F}$ such that a cost function also known as average distortion

$$E(M, \mathcal{Y}, \mathcal{X}) = \frac{1}{N} \sum_{j=1}^{K} \sum_{i=1}^{N} m_{ij} \delta(\boldsymbol{x}_i, \boldsymbol{y}_j) \tag{2}$$

is minimized with respect to a given *distortion measure* δ. The matrix $M = (m_{ij}) \in \mathcal{M}_{N \times K}(\{0, 1\})$ is a binary *membership matrix* with the constraints

$$\sum_{j=1}^{K} m_{ij} = 1 \tag{3}$$

for all $i = 1, \ldots, N$. The constraints assure that each data point is represented by a unique model. Fuzzy clustering or topology preserving clustering methods like self-organizing feature maps relax the hard constraints on the membership matrix M and demand entries m_{ij} in the range of [0, 1] subject to (3).

The choice of an appropriate distortion measure δ depends on the particular application domain. A common choice of δ for *K-means clustering* is of the form

$$\delta(\boldsymbol{x}_i, \boldsymbol{y}_j) = \|\boldsymbol{x}_i - \boldsymbol{y}_j\|^2.$$

Alternatively, the clustering task can be formulated as a maximization task where the distortion δ is given by a similarity measure.

The cost function can be minimized by various deterministic or stochastic methods from combinatorial and continuous optimization. We distinguish between two iterative methods

```
The $K$-Means Clustering Algorithm:
Let
  – $\mathcal{X} = \{\boldsymbol{x_1}, \dots, \boldsymbol{x_N}\} \subseteq \mathbb{R}^d$ be a set of data points
  – $M = (m_{ij}) \in \mathcal{M}_{N \times K}(\{0, 1\})$ be the membership matrix.
 1. Initialize $K$ and $\mathcal{Y} = \{\boldsymbol{y_1}, \dots \boldsymbol{y_K}\} \subseteq \mathbb{R}^d$.
 2. Repeat
    a) Set $m_{ij} = 0$ for all $i, j$
    b) For all $i = 1, \dots, N$:
       i) Classify $\boldsymbol{x_i} \in \mathcal{X}$ according to nearest neighbor rule
              $i^* = \arg\min_j \delta(\boldsymbol{x_i}, \boldsymbol{y_j})$.
       ii) Set $m_{ii^*} = 1$.
    c) For all $j = 1, \dots, K$: Recompute means
              $y_j = \frac{1}{n_j} \sum_i m_{ij} x_i$
       where $n_j = \sum_i m_{ij}$.
 3. until no noticeable changes in $\mathcal{Y}$ are observed.
 4. Return cluster centers $\mathcal{Y} = \{\boldsymbol{y_1}, \dots \boldsymbol{y_K}\}$.
```

Figure 1. Outline of the K-means algorithm.

of cluster optimization. Given a data point $\boldsymbol{x_i}$, *hard optimizers* only adjust the cluster center $\boldsymbol{y}_{i^*}$ nearest to $\boldsymbol{x_i}$ with respect to the distortion measure δ. Examples for hard optimizers are the batch and online version of the *K-means* algorithm or *simple competitive learning* (Palmer Hertz, & Krogh, 1991). The second approach to minimize the clustering cost function, referred to as *soft optimizers*, iteratively adjust all cluster centers $\boldsymbol{y}_j$ for which the current assignment $m_{ij} > 0$. This class of optimizers include, for example, self-organizing feature maps or probabilistic partitioning algorithms.

To illustrate the main ideas of our exposition, we simply choose the K-Means algorithm as a representative hard optimizer for the clustering cost function (2) as outlined in figure 1.

4. Central clustering of attributed graphs

Clustering N structured objects which are represented by attributed graphs $\mathcal{X} = \{X_1, \dots, X_M\} \subseteq \mathcal{G}$ amounts to partition the feature space $\mathcal{G}$ such that the average distortion of data graphs to their cluster centers $\mathcal{Y} = \{Y_1, \dots, Y_K\} \subseteq \mathcal{G}$ is minimized. As in Section 3, the average distortion to be minimized is of the form

$$E(M, \mathcal{Y}, \mathcal{X}) = \frac{1}{N} \sum_{j=1}^{K} \sum_{i=1}^{N} m_{ij} \delta(X_i, Y_j) \tag{4}$$

where δ measures the structural distortion induced by the representation Y_{i^*} of data graph X_i. This section extends the K-means algorithm from the domain of real valued feature vectors to the domain of attributed graphs.

To utilize the K-means algorithm for structured objects two basic questions at issue must be addressed:

1. How can we adjust a cluster center in the domain of attributed graphs as in Step (2c) of the K-means algorithm. Section 4.1 is concerned with this question at issue.
2. The second question at issue arises from a practical point of view. One epoch (cycle through the training set) of the K-means algorithm requires the computation of $N \cdot K$ distortions. The time complexity of determining structural distortions of attributed graphs is NP-complete. Also approximating structural distortions is a computational intensive procedure. Thus, even though we know how to adjust the cluster centers, applying the K-means algorithm in an unmodified form to attributed graphs might be intractable in a practical setting. Section 4.2 suggest a solution to this issue.

4.1. Cluster optimization in a metric graph space

In Section 4.1.1 we define a metric graph space to establish a theoretical foundation for central clustering. A metric graph space adopts properties of metric vector space required for adjusting the cluster centers to minimize the cost function (4). For this purpose we embed the domain of graphs $X \in \mathcal{G}^n$ into an inner product space $\mathcal{A}^{n \cdot n}$ by means of their adjacency matrices $A(X) \in \mathcal{M}_{n \times n}(\mathcal{A})$ without loss of structural information. A degenerated form of an inner product is then defined on the adjacency matrices taking into account the special numbering of the vertex sets. All other necessary requisites to apply cluster optimization algorithms can be derived from the inner product under consideration. Note that the theoretical foundation given in Section 4.1.1 is independent from the particular cluster optimization algorithm. Section 4.1.2 exemplary illustrates how the properties of a metric graph space can be utilized to formulate a generic k-means algorithm for structures (KMS).

4.1.1. Metric graph spaces. The high computational effort in measuring the proximity of two given graphs arises from the large number of possible combinatorial arrangements of the vertices to represent the same abstract graph. Thus any reasonable proximity measure in the domain of graphs is independent of the particular labeling and rather evaluates either common or distinct structural properties. In mathematical terms, a proximity measure $\xi : \mathcal{G} \times \mathcal{G} \to \mathbb{R}$ should satisfy

$$X \simeq X' \Rightarrow \xi(X, Y) = \xi(X', Y) \tag{5}$$

for all $X, X', Y \in \mathcal{G}$.

In order to adopt metric and arithmetic properties of real valued vectors for graphs, such that (5) is satisfied, we embed the domain of graphs of order n into the inner product space space $\mathcal{A}^{n\cdot n}$ by means of their adjacency matrices. Addition, scalar multiplication, the scalar product, and the Euclidean distance of graphs are then defined in terms of their adjacency matrices taking into account the particular labeling of the vertices.

We start our treatment with some technical conventions and definitions:

1. In order to facilitate arithmetic operations on graphs by means of their adjacency matrices we align graphs of different order to graphs of equal order as follows: Graphs $X \in \mathcal{G}^n$ of order $|X| = p < n$ are aligned to a graph $X' \in \mathcal{G}^n$ by inserting $q = n - p$ vertices of weight zero such that the adjacency matrix $A(X') \in \mathcal{M}_{n\times n}(\mathcal{A})$ of X' is of the form

$$A(X') = \begin{pmatrix} A(X) & \mathbf{0}_{pq} \\ \mathbf{0}_{qp} & \mathbf{0}_{qq} \end{pmatrix}$$

   where $A(X)$ is the adjacency matrix of X and $\mathbf{0}_{rs} \in \mathcal{M}_{r\times s}(\mathcal{A})$ is a matrix with identical entries $\mathbf{0} \in \mathcal{A}$. Note, that a structure preserving alignment of graphs with respect to a fixed order n does not confine the following considerations to graphs of bounded order. It is rather a pure auxiliary construct to simplify mathematical technicalities.
2. Let $A = (a_{ij}), B = (b_{ij}) \in \mathcal{M}_{n\times n}(\mathcal{A})$ and $\lambda \in \mathbb{R}$. Then addition and scalar multiplication of matrices are defined component-wise

$$\begin{aligned}(a_{ij}) + (b_{ij}) &= (a_{ij} + b_{ij})\\ \lambda \cdot (a_{ij}) &= (\lambda \cdot a_{ij})\end{aligned}$$

3. Let $A, B \in \mathcal{M}_{n\times n}(\mathcal{A})$. The inner product $\langle\,,\rangle$ associated with $\mathcal{A}^{n\cdot n}$ induces an inner product on $\mathcal{M}_{n\times n}(\mathcal{A})$ by

$$\langle A, B\rangle = \langle \mathrm{vec}(A), \mathrm{vec}(B)\rangle$$

   where $\mathrm{vec}(A), \mathrm{vec}(B) \in \mathcal{A}^{n\cdot n}$ are the vectors obtained by concatenating the rows of A and B, respectively.
4. Let $\|\cdot\|$ be a norm defined on $\mathcal{A}^{n\cdot n}$. Then $\|A\| = \|\mathrm{vec}(A)\|$ is a norm on $\mathcal{M}_{n\times n}(\mathcal{A})$.
5. Let $X, Y \in \mathcal{G}^n$ be attributed graphs and $\lambda \in \mathbb{R}$. Then

$$\begin{aligned}X + Y &= A(X) + A(Y)\\ \lambda \cdot X &= \lambda \cdot A(X)\end{aligned}$$

   Note, that in general $X + Y \neq X' + Y$ for isomorphic graphs $X \simeq X'$. In this general form, the sum of two graphs is a pure syntactical construct without any graph theoretical meaning.

Fundamental for the following considerations and derivations is the *Schur-Hadamard inner product* of pairs of graphs in $\mathcal{G}^n$. The concept of a Schur-Hadamard inner product of graphs can be regarded as a degenerated counterpart of the concept of an inner product defined on vector spaces.

Definition 4.1 (Schur-Hadamard inner product on $\mathcal{G}^n$). Let $\mathcal{A}^{n \cdot n}$ be an inner product space over $\mathbb{R}$ with associated inner product $\langle\, ,\, \rangle$. Then

$$\sigma : \mathcal{G}^n \times \mathcal{G}^n \to \mathbb{R}, \quad (X, Y) \mapsto \max_{\pi \in \mathcal{S}_n} \langle A(X^\pi), A(Y) \rangle$$

is the Schur-Hadamard inner product induced by $\langle\, ,\, \rangle$.

From the definition of the SH inner product directly follows that

$$\sigma(X, Y) \geq \langle A(X^\pi), A(Y) \rangle \tag{6}$$

for all $\pi \in \mathcal{S}_n$. Permutations $\pi \in \mathcal{S}_n$ for which equality holds in (6) are the *embeddings* from X into Y. By $\mathcal{I}(X, Y)$ we denote the set of all embeddings from X into Y.

The SH inner product allows a geometrical interpretation. Any graph can be embedded in the inner product space $\mathcal{A}^{n \cdot n}$ as a set of vectors $\mathcal{V}_X = \{\text{vec}(A(X^\pi)) : \pi \in \mathcal{S}_n\}$. The cardinality of $\mathcal{V}_X$ depends on the number of automorphisms, that is the number of permutations $\pi \in \mathcal{S}_n$ which act invariantly on the adjacency matrix of X. The SH inner product of X and Y is the inner product of two vectors $\boldsymbol{x} \in \mathcal{V}_X$ and $\boldsymbol{y} \in \mathcal{V}_Y$ with smallest enclosing angle.

The following result lists some properties of the SH inner product, which will be used implicitly throughout this section.

Proposition 4.1. *Let σ be a Schur-Hadamard inner product defined on $\mathcal{G}^n$. Then the following properties are satisfied for all $X, Y, Z \in \mathcal{G}^n$*
SH 1. $\sigma(X, X) = \langle A(X), A(X) \rangle \geq 0$
SH 2. $\sigma(X, X) = 0 \Leftrightarrow A(X) = 0_{n,n}$
SH 3. $\sigma(X, Y) = \sigma(Y, X)$
SH 4. $\sigma(\lambda \cdot X, Y) = \lambda \cdot \sigma(X, Y)$ *for all* $\lambda \in \mathbb{R}_+$
SH 5. $\sigma(X + Y, Z) \leq \sigma(X, Z) + \sigma(Y, Z)$

Proof: (SH 1): Let $\pi \in \mathcal{I}(X, X)$ be an embedding. Then

$$\begin{aligned} \sigma(X, X) &= \langle A(X^\pi), A(X) \rangle \\ &= \|A(X^\pi)\| \cdot \|A(X)\| \cdot \cos \alpha \\ &= \langle A(X), A(X) \rangle \cdot \cos \alpha \\ &\leq \langle A(X), A(X) \rangle \end{aligned}$$

where α is the angle between $vec(A(X^\pi))$ and $vec(A(X))$. The equation in the third line follows from $\|A(X)\| = \|A(X^\pi)\|$ for all $\pi \in \mathcal{S}_n$. Since π is an embedding, we have $\langle A(X^\pi), A(X)\rangle \geq \langle A(X), A(X)\rangle$. Combining both inequalities yields $\sigma(X, X) = \langle A(X), A(X)\rangle$. This together with $\langle A(X), A(X)\rangle \geq 0$ proves property (SH 1).

(SH 2): Follows directly from the positive definiteness of $\langle\,,\,\rangle$.

(SH 3): Assume that $\pi \in \mathcal{I}(X, Y)$ is an embedding from X into Y. Then by definition of the Schur-Hadamard-inner product induced by $\langle\,,\,\rangle$ we have

$$\sigma(X, Y) = \langle A(X^\pi), A(Y)\rangle$$

Since $\pi \in \mathcal{S}_n$ there exists a permutation matrix $P \in \mathcal{M}_{n\cdot n\times n\cdot n}(\{0, 1\})$, such that

$$A(X^\pi) = P^t A(X) P$$

Permutation matrices are orthogonal and preserve length. Repeated application of the properties of an orthogonal matrix yields

$$\begin{aligned}\sigma(X, Y) &= \langle P^t A(X)P, A(Y)\rangle\\ &= \langle P(P^t A(X)P), PA(Y)\rangle\\ &= \langle (A(X)P)P^t, (PA(Y))P^t\rangle\\ &= \langle A(X), PA(Y)P^t\rangle = \langle PA(Y)P^t, A(X)\rangle\end{aligned}$$

where the last equation follows from the symmetry of the inner product. From $PA(Y)P^t = A(Y^\phi)$ with $\phi = \pi^{-1} \in \mathcal{S}_n$ follows that $\sigma(X, Y) \leq \sigma(Y, X)$. A similar argumentation shows $\sigma(Y, X) \leq \sigma(X, Y)$. Combining both inequalities yields property (SH 3).

(SH 4): Let $\pi \in \mathcal{I}(X, Y)$ be an embedding and $\lambda \in \mathbb{R}_+$. It is sufficient to show that $\pi \in \mathcal{I}(\lambda \cdot X, Y)$. Since $\pi \in \mathcal{I}(X, Y)$ is an embedding, it maximizes the Schur-Hadamard inner product of adjacency matrices

$$\langle A(X^\pi), A(Y)\rangle = \max_{\phi\in\mathcal{S}_n} \langle A(X^\phi), A(Y)\rangle\ .$$

From $\lambda \geq 0$ and

$$\lambda \cdot \max_{\phi\in\mathcal{S}_n} \langle A(X^\phi), A(Y)\rangle = \max_{\phi\in\mathcal{S}_n} \langle \lambda \cdot A(X^\phi), A(Y)\rangle$$

follows $\pi \in \mathcal{I}(\lambda \cdot X, Y)$.

(SH 5): Let $\pi \in \mathcal{I}(X + Y, Z)$, $\phi \in \mathcal{I}(X, Z)$, and $\psi \in \mathcal{I}(Y, Z)$ be embeddings into Z. Then

$$\begin{aligned}\sigma(X + Y, Z) &= \langle A((X + Y)^\pi), A(Z)\rangle = \langle A(X^\pi) + A(Y^\pi), A(Z)\rangle\\ &= \langle A(X^\pi), A(Z)\rangle + \langle A(Y^\pi), A(Z)\rangle\\ &\leq \langle A(X^\phi), A(Z)\rangle + \langle A(Y^\psi), A(Z)\rangle = \sigma(X, Z) + \sigma(Y, Z)\ .\end{aligned}$$

□

The SH inner product gives rise to the *Euclidean length* or *norm* of an attributed graph X

$$\|X\| = \sqrt{\sigma(X, X)}.$$

According to (SH 1) the length of a graph X can simply be computed by taking the square root of the inner product $\langle A(X), A(X)\rangle$. Although the SH inner product is in fact not an inner product, the norm of an attributed graph induces a metric on $\mathcal{G}^n$ in the obvious way.

Theorem 4.1. *Let* $\| \ \|$ *be the Euclidean length on* $\mathcal{G}^n$. *Then the mapping*

$$\delta : \mathcal{G}^n \times \mathcal{G}^n \to \mathbb{R}, \quad (X, Y) \mapsto \min_{\pi \in \mathcal{S}_n} \|X^\pi - Y\|$$

is a distance metric on $\mathcal{G}^n$.

Proof: First we show that $\delta(X, Y) = \|X^\pi - Y\|$ if and only if $\pi \in \mathcal{I}(X, Y)$. We have

$$\begin{aligned}\delta(X, Y)^2 &= \min_{\pi \in \mathcal{S}_n} \|X^\pi - Y\|^2 \\ &= \min_{\pi \in \mathcal{S}_n} \langle A(X^\phi) - A(Y), A(X^\phi) - A(Y)\rangle \\ &= \min_{\pi \in \mathcal{S}_n} \{\|X\|^2 - 2\langle A(X^\phi), A(Y)\rangle + \|Y\|^2\} \\ &= \|X\|^2 - 2 \max_{\pi \in \mathcal{S}_n} \langle A(X^\phi, A(Y)\rangle + \|Y\|^2 \\ &= \|X\|^2 - 2\sigma(X, Y) + \|Y\|^2.\end{aligned}$$

Next we prove that δ satisfies the properties of a metric. Symmetry of δ follows from the first part of this proof together with the symmetry of σ.

Now we show that $\delta(X, Y) = 0$ if and only if $X \simeq Y$. Clearly, $\delta(X, X) = 0$. Let $\pi \in \mathcal{I}(X, Y)$ be an embedding. From

$$\delta(X, Y) = \|X^\pi - Y\| = \|A(X^\pi) - A(Y)\| = 0$$

follows $A(X^\pi) = A(Y)$ and therefore $X^\pi = Y$.

Finally we show that δ satisfies the triangle inequality. Let $\pi \in \mathcal{I}(X, Y)$, $\psi \in \mathcal{I}(Z, Y)$, and $\phi \in \mathcal{I}(X, Z^\psi)$ be embeddings. Since ϕ is not necessarily an embedding from X to Y, we have

$$\begin{aligned}\delta(X, Y) &= \|X^\pi - Z^\psi + Z^\psi - Y\| \\ &\leq \|X^\phi - Z^\psi + Z^\psi - Y\| = \|A(X^\phi) - A(Z^\psi) + A(Z^\psi) - A(Y)\|.\end{aligned}$$

Applying the triangle inequality of the norm $\|.\|$ defined on $\mathcal{A}^{n\cdot n}$ yields

$$\begin{aligned}\|X^{\phi} - Z^{\psi} + Z^{\psi} - Y\| &= \|A(X^{\phi}) - A(Z^{\psi}) + A(Z^{\psi}) - A(Y)\| \\ &\leq \|A(X^{\phi}) - A(Z^{\psi})\| + \|A(Z^{\psi}) - A(Y)\| \\ &= \|X^{\phi} - Z^{\psi}\| + \|Z^{\psi} - Y\|.\end{aligned}$$

From $\phi \in \mathcal{I}(X, Z^{\psi})$ and $\psi \in \mathcal{I}(Z, Y)$ follows

$$\|X^{\phi} - Z^{\psi}\| + \|Z^{\psi} - Y\| = \delta(X, Z) + \delta(Z, Y).$$

Putting all together we have proven the triangle inequality. □

We call the distance metric δ defined in Theorem 4.1 *Euclidean metric with respect to* σ or simply *Euclidean metric*. The proof of Theorem 4.1 supplies two properties of the Euclidean metric summarized in Corollary 4.1.

Corollary 4.1. *Let $X, Y \in \mathcal{G}^n$ be attributed graphs. Then*
1. $\delta(X, Y) = \|X^{\pi} - Y\| \;\Leftrightarrow\; \pi \in \mathcal{I}(X, Y)$
2. $\delta(X, Y) = \sqrt{\|X\|^2 - 2\sigma(X, Y) + \|Y\|^2}$

Under the assumption that the graphs are normalized we can show that maximizing the SH inner product of two given graphs is equivalent to minimizing their Euclidean distance.

Corollary 4.2. *Let $X, Y, Z \in \mathcal{G}^n$ be attributed graphs with $\|Y\| = \|Z\|$.*
1. $\sigma(X, Y) \geq \sigma(X, Z) \;\Leftrightarrow\; \delta(X, Y) \leq \delta(X, Z)$
2. $\langle A(X^{\phi}), A(Y)\rangle > \langle A(X^{\psi}), A(Y)\rangle \;\Rightarrow\; \|X^{\phi} - Y\| < \|X^{\psi} - Y\|$

Proof: Follows directly from the second property of Corollary 4.1. □

Next we define the *weighted mean* M of graphs X and Y. The weighted mean is useful to extend central clustering methods like self-organizing feature maps. In addition it can be used to approximate a sample mean of a set of data graphs.

Definition 4.2 (Weighted mean). Let $X, Y \in \mathcal{G}^n$ be graphs, $\pi \in \mathcal{I}(X, Y)$ an embedding, and $\eta \in [0, 1]$ be a constant. Any graph $M \in \mathcal{G}^n$ with

$$M \simeq \eta X^{\pi} + (1 - \eta)Y. \tag{7}$$

is a *weighted mean* of X and Y.

Theorem 4.2 shows that M is indeed a weighted mean of X and Y in an Euclidean sense.

Theorem 4.2. *Let $\pi \in \mathcal{I}(X, Y)$ an embedding from X into Y and $M = \eta X^{\pi} + (1 - \eta)Y$ be a weighted mean of graphs X and Y. Then the following equations hold*

$$\begin{aligned}\delta(X, M) &= |1 - \eta| \cdot \delta(X, Y)\\ \delta(M, Y) &= |\eta| \cdot \delta(X, Y)\end{aligned}$$

Proof: We only show the first equation. The argumentation for the second equation is similar. First we show that $\pi \in \mathcal{I}(X, M)$ is an embedding from X into M. For this purpose it is sufficient to show that π maximizes the inner product $\langle A(X^{\pi}), A(M)\rangle$. Since $M = \eta X^{\pi} + (1 - \eta)Y$ we have

$$A(M) = \eta A(X^{\pi}) + (1 - \eta)A(Y).$$

This gives

$$\begin{aligned}\langle A(X^{\pi}), A(M)\rangle &= \langle A(X^{\pi}), \eta A(X^{\pi}) + (1 - \eta)A(Y)\rangle\\ &= \eta \cdot \langle A(X^{\pi}), A(X^{\pi})\rangle + (1 - \eta) \cdot \langle A(X^{\pi}), A(Y)\rangle.\end{aligned}$$

According to (SH 1) the first term in the second line is exactly $\sigma(X^{\pi}, X^{\pi})$ and thus maximal. By assumption π is an embedding from X into Y and thus the second term is also maximal. Since both terms are maximal, the sum is maximal. Hence π is an embedding from X into M. Now we are able to show the first equation. We have

$$\begin{aligned}\delta(X, M) &= \|X^{\pi} - M\| = \|X^{\pi} - (\eta X^{\pi} + (1 - \eta)Y)\|\\ &= |1 - \eta| \cdot \|X^{\pi} - Y\| = |1 - \eta| \cdot \delta(X, Y).\end{aligned}$$

□

Note that a weighted mean of two graphs is in general not uniquely determined. Next we adopt the notion of sample mean for attributed graphs. This concept is useful to formulate central clustering algorithms like K-means. The sample mean of a set of vectors in an Euclidean vector space can be alternatively expressed in terms of the Euclidean distance as the vector having the smallest sum of distances over the data set. We use this equivalent definition to define a sample mean of attributed graphs. For that purpose let

$$E : \mathcal{G}^n \to \mathbb{R}, \quad Y \mapsto \sum_{i=1}^{N} \delta(X_i, Y). \tag{8}$$

denote the function measuring the total distortion between a given attributed graph Y and the samples X_i with respect to δ.

Definition 4.3 (Sample mean). A *sample mean* of N graphs $X_1, \ldots, X_N \in \mathcal{G}^n$ is a graph $M \in \mathcal{G}^n$ with

$$M = \arg \min_{Y \in \mathcal{G}^n} E(Y). \tag{9}$$

As the weighted mean, the sample mean is, in general, not uniquely determined. The following Theorem shows that a sample mean of a set of attributed graphs has a similar form as its counterpart in an Euclidean vector space.

Theorem 4.3. *Let M be a sample mean of $X_1, \ldots, X_N \in \mathcal{G}^n$. Then there are permutations $\phi_1, \ldots, \phi_N \in \mathcal{S}_n$ such that*

$$M \simeq \frac{1}{N} \sum_{i=1}^{N} X_i^{\phi_i}. \tag{10}$$

Proof: Let

$$\tilde{E} : \mathcal{G}^n \to \mathbb{R}, \quad Y \mapsto \frac{1}{2} \sum_{i=1}^{N} \delta(X_i, Y)^2$$

be a function. Clearly, minimizing $\tilde{E}$ is equivalent to minimizing E given in (8). By $\tilde{E}_*$ we denote the minimum of $\tilde{E}$. Since M is a sample mean, there are embeddings $\phi_i \in \mathcal{I}(X_i, M)$ for all $1 \leq i \leq N$ such that

$$\tilde{E}_* = \sum_{i=1}^{N} \left\| X_i^{\phi_i} - M \right\|^2.$$

For fixed embeddings ϕ_i, the function $\tilde{E}$ is continuous and differentiable in Y. Thus a necessary condition for $\tilde{E}$ to have a minimum at $Y = M$ is that

$$\frac{d\tilde{E}}{dY} = -2 \sum_{i=1}^{N} X_i^{\phi_i} - Y = 0. \tag{11}$$

Since $\tilde{E}$ is convex, we find that M is a global minimum of $\tilde{E}$. Solving Eq. (11) for Y yields the assumption. □

4.1.2. A generic K-means clustering algorithm for structures. Given the notion of a metric graph space, we are now able to extend central clustering algorithms like the K-means algorithm, simple competitive learning, self-organizing feature maps, or probabilistic partitioning algorithms from the domain of feature vectors to the domain of attributed graphs.

In this subsection, we exemplary illustrate how to utilize the results of 4.1.1 to apply central clustering methods to the domain of attributed graphs. The chosen example is the well known and simple K-means algorithm.

Let $\mathcal{X} = \{X_1, \dots, X_N\}$ be a set of data graphs and $\mathcal{Y} = \{Y_1, \dots, Y_N\}$ the set of model graphs. Then the essential elements of K-means are assignment of the patterns X_i to their optimal reference models Y_{i^*} and recomputation of the models Y_j given the new assignments of the data. To assign a data graph X_i to its optimal reference model we use the nearest neighbor rule

$$Y_{i^*} = \arg\min_{Y_j \in \mathcal{Y}} \delta(X_i, Y_j) \tag{12}$$

where δ is the Euclidean distance of attributed graphs. After each data graph is assigned to its current optimal reference model, we may recompute model Y_j in accordance with Step (2c) of the K-means algorithm for feature vectors as outlined in figure 1. Assume that $X_{i_1}, \dots, X_{i_{n_j}}$ are assigned to model Y_j. By Theorem 4.3 a recomputed model Y_j is of the form

$$Y_j \simeq \frac{1}{n_j} \sum_{k=1}^{n_j} X_{i_k}^{\phi_{i_k}}$$

where $\Phi_j = \{\phi_{i_1}, \dots, \phi_{i_{n_j}}\} \subseteq \mathcal{S}_n$ are permutations such that

$$Y_j = \arg\min_{Y \in \mathcal{G}^n} \sum_{k=1}^{n_j} \left\| X_{i_k}^{\phi_{i_k}} - Y \right\|.$$

Unfortunately, all we know about Φ_j is that each $\phi_{i_k} \in \Phi_j$ is an embedding $\phi_{i_k} \in \mathcal{I}(X_{i_k}, M_{i_k})$ where

$$M_{i_k} = \sum_{l \neq k} X_{i_l}^{\phi_{i_l}}$$

is the sum of the samples $X_{i_l}^{\phi_{i_l}}$ without $X_{i_k}^{\phi_{i_k}}$. The search space of all possible combinations $(\phi_{i_1}, \dots \phi_{n_j}) \in \mathcal{S}_n^{n_j}$ consists of $(n!)^{n_j}$ elements, which is exponential in both, the number of vertices of the X_{i_k} and the cardinality n_j of the set of samples assigned to model Y_j. Thus in a practical setting we have to resort to approximations of a sample mean, because determining permutations $\phi_{i_1}, \dots, \phi_{i_{n_j}}$ which minimize the total distortion (8) is of very large computational complexity. An experimental evaluation revealed that computation of a sample mean is prohibitively time consuming even for a small number of graphs. One way around this problem is a greedy-like approximation of a sample mean, called *incremental mean*.

Definition 4.4 (Incremental mean). Let $X_1, \dots, X_N \in \mathcal{G}^n$ be attributed graphs. An *incremental mean* of $X_1, \dots, X_N$ is a graph $M \simeq M_N \in \mathcal{G}^n$ recursively defined by

$$\begin{aligned} M_1 &= X_1 \\ M_k &= \frac{k-1}{k} M_k + \frac{1}{k} X_k^{\pi_k} \quad (1 < k \leq N) \end{aligned}$$

where $\pi_k \in \mathcal{I}(X_k, M_k)$ is an embedding.

An incremental mean M is not unique, because M depends on the ordering of the data graphs X_i. Even if the ordering of the data graphs is assumed to be fixed, M is in general not uniquely determined. The reason is that $M \simeq M_N$ and each M_k is a weighted mean of M_{k-1} and X_k for all $1 < k \leq N$. From the non-uniqueness of a weighted mean directly follows that an ordered incremental mean is not uniquely determined as well.

An incremental mean approach as an approximator of a sample mean is motivated by the following consideration: Any minimizer Y of the total distortion (8) matches the samples X_i of a data set $\mathcal{X}$ as largely as possible. A large match between Y and a sample X_i corresponds to a large similar common substructure of both graphs. On the other hand an incremental mean of $\mathcal{X}$ iteratively averages over common substructures of the elements of $\mathcal{X}$. The larger the averaged common substructures the less the total distortion (8) between an incremental mean and the graphs of $\mathcal{X}$. Putting both observations together suggests to use an incremental mean as an approximation of a sample mean.

Figure 2 summarizes the generic KMS algorithm, generic in the sense that the given formulation is independent of the particular algorithm to determine the distortions $\delta(X, Y_i)$ and the embeddings $\pi \in \mathcal{I}(X, Y_{i^*})$.

```
A Generic $K$-Means Algorithm for Structures:
Let $\mathcal{X} = \{X_1, \dots, X_N\} \subseteq \mathcal{G}^n$ be a set of data graphs.
  1. Initialize $K$ and $\mathcal{Y} = \{Y_1, \dots Y_K\} \subseteq \mathcal{G}^n$.
  2. Repeat
      a) For all $i = 1, \dots, N$:
             Classify $X_i \in \mathcal{X}$ according to nearest neighbor rule
                 $i^* = \arg\min_j \delta(X_i, Y_j)$.
      b) For all $j = 1, \dots, K$:
             Recompute incremental mean $Y_j$
  3. until no noticeable changes in $\mathcal{Y}$ are observed.
  4. Return cluster centers $\mathcal{Y} = \{\boldsymbol{y_1}, \dots \boldsymbol{y_K}\}$.
```

Figure 2. Outline of the generic KMS algorithm.

4.2. A neural network approach for the generic KMS *algorithm*

In Section 4.1 we proposed a generic KMS algorithm. This algorithm first classifies all samples $X_i \in \mathcal{X}$ of a set of data graphs according to the nearest neighbor rule as shown in figure 2, Step (2a). Then each model $Y_j \in \mathcal{Y}$ is adjusted according to the learning rule given in figure 2, Step (2b).

The usual method to classify X is to first compute all K distortions $\delta(X, Y_i)$. Then a comparative *minimum selector* (MS) identifies the optimal reference model Y_{i^*} of X with respect to the nearest neighbor rule. The MS classifier approach is simple and easy to implement in the domain of real valued feature vectors. In addition it guarantees to return an optimal model with respect to the chosen distortion measure δ. In the domain of graphs, however, determining a structural distortion of two given graphs is NP-complete (Garey & Johnson, 1979). Due to the exponential computational complexity of the graph matching problem, exact algorithms which guarantee to return an optimal solution are useless in a practical clustering task for all but the smallest graphs. Therefore, we abandon the guarantee of selecting the optimal reference model for a given data graph and resort to heuristics which approximately solve the graph matching problem within an acceptable time limit. Due to its applicability in various relevant application areas, many heuristics have been devised to approximately solve the graph matching problem (Christmas, Kittler, & Petrou, 1995; Cross, Wilson, & Hancock, 1997; Eshera & Fu, 1986; Gold & Rangarajan, 1996; Herault et al., 1990; Pelillo, Siddiqi, & Zucker, 1999; Rangarajan, Gold, & Mjolsness, 1996; Schädler & Wysotzki, 1999; Suganthan, Teoh, & Mital, 1995; Shapiro & Haralick, 1981; Williams, Wilson, & Hancock, 1999; Wong & You, 1985; Yu & Tsai, 1992). Nevertheless, even approximating structural distortions of attributed graphs is a computational intensive procedure compared to the time complexity of determining, for example, the Minkowski distance of real valued vectors. Thus when clustering structured objects via an MS classification procedure, one iteration step of the KMS algorithm requires K computational intensive approximations of the distortion measure induced by the current data graph and the K models. Therefore clustering structured objects can still remain computational intractable in a practical setting, in particular for huge data sets of large graphs.

In this Section we propose a neural network implementation of the KMS algorithm. We replace the minimum selector rule of an MS classifier by the principle of successive *elimination of competition* by means of an *extended winner-takes-all* (X-WTA) network. The X-WTA architecture has been first introduced in Jain and Wysotzki (2001, 2002) and since then enhanced in Jain and Wysotzki (in press, 2003a, 2003b). Here we customize the X-WTA architecture for the KMS algorithm. The basic procedure of X-WTA network simultaneously matches the input data X with the K models Y_i using K Hopfield networks $\mathcal{H}_i$, called *subnets*. Following the principle *elimination of competition* the X-WTA network focuses on promising subnets and disables unfavorable subnets until one subnet wins the competition and all other subnets are disabled. The input X is assigned to the model corresponding to the winner. As opposed to the minimum selector rule of an MS classifier, the X-WTA procedure circumvents the computational intensive approximations of K distortions and selects a winner without completing the computation of a single subnet.

In Section 4.2.1 we describe a special Hopfield network for approximately solving the graph matching problem. Section 4.2.2 deals with inhibitory winner-takes-all networks for maximum selection. In Section 4.2.3 we propose the X-WTA network. Finally, Section 4.3 assembles the individual parts to an KMS algorithm.

4.2.1. Graph matching with neural networks. In this subsection we describe how to compute the Schur-Hadamard inner product of two attributed graphs A and Y using neural networks. From the Schur-Hadamard inner product we can simply derive the Euclidean distance of X and Y (Corollary 4.1). Thus this approach is also able to deal with clustering methods like simple competitive learning which select the best matching model of a given input with respect to the maximal inner product.

Following the seminal paper of Hopfield and Tank (1985), the general approach to solve combinatorial optimization problems by using neural networks maps the objective function of the optimization problem onto an energy function of the network. The constraints of the problem are included in the energy function as penalty terms, such that the global minima of the energy function correspond to the solutions of the combinatorial optimization problem.

Here the combinatorial optimization problem is the problem of finding an embedding. Thus by definition of an embedding the objective function to maximize is the SH inner product. To solve that problem we introduce a neural network approach, which is derived from the maximum clique detection in an *association graph* (Ballard & Brown, 1982). The classical association graph P of two graphs X and Y is an auxiliary structure coding each (maximal) common induced subgraph of X and Y into a (maximal) clique of P. This maps the search for a maximum common induced subgraph of X and Y to the problem of finding a maximum clique in P. Since a maximum common subgraph and an embedding are similar concepts in the sense that they both describe a structural overlap of X and Y, we shall customize the maximum clique approach to the problem of finding an embedding from X into Y.

Maximum weighted cliques and the SH *inner product*

This section transforms the problem of computing the SH inner product to the problem of finding a maximum weighted clique in an inner product graph. A detailed description of this approach is given in Jain and Wysotzki (in press).

To simplify technicalities and to keep the main ideas clear we restrict the following consideration to the canonical SH inner product represented by the identity matrix.

Inner product graphs

Let X, $Y \in \mathcal{G}^n$ be graphs with adjacency matrices $A(X) = (x_{ij})$ and $A(Y) = (y_{kl})$, respectively. For any pair $\boldsymbol{i} = (i, k)$, $\boldsymbol{j} = (j, l)$ of elements of the Cartesian Product $V(X) \times V(Y)$, the symbol $\chi_{\boldsymbol{ij}}$ denotes the *compatibility index* of $\boldsymbol{i}$ and $\boldsymbol{j}$

$$\chi_{\boldsymbol{ij}} = \begin{cases} 1 & \text{if } (i \neq j \wedge k \neq l) \vee (i = j \wedge k = l) \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

If we regard each pair $\boldsymbol{i} = (i, k)$ as a mapping $i \mapsto k$ of vertex $i \in V(X)$ to vertex $k \in V(Y)$, then two pairs $\boldsymbol{i} = (i, k)$ and $\boldsymbol{j} = (j, l)$ are compatible if and only if the union of the mappings $i \mapsto k$ and $j \mapsto l$ is bijective.

The *inner product graph* $Z = X \Diamond_N Y$ of X and Y is a weighted graph defined on the Cartesian Product $V(Z) = V(X) \times V(Y)$. The edge set

$$E(Z) = \{(\boldsymbol{i}, \boldsymbol{j}) \in V(Z) : \chi_{\boldsymbol{ij}} = 1\}.$$

is the set of all pairs $(\boldsymbol{i}, \boldsymbol{j})$ of vertices from $V(Z)$ which are compatible. The weights of Z are

$$\mu_Z(\boldsymbol{i}, \boldsymbol{j}) = \chi_{\boldsymbol{ij}} \cdot N(\langle x_{ij}, y_{kl}\rangle) \tag{14}$$

for all pairs of vertices $\boldsymbol{i} = (i, k)$ and $\boldsymbol{j} = (j, l)$ of Z. The function $N(\cdot)$ is a bijective linear transformation such that $0 < \mu_Z(\boldsymbol{i}, \boldsymbol{j}) \leq 1$ for all $\boldsymbol{i}, \boldsymbol{j} \in V(Z)$ with $\chi_{\boldsymbol{ij}} = 1$.

Note that, although $N(\cdot)$ is not uniquely determined, the structure of the inner product graph is independent of the particular choice of $N(\cdot)$. The transformation $N(\cdot)$ yields a weighted graph Z such that all vertices and edges of Z have positive weight. Non-edges of Z correspond to pairs $(\boldsymbol{i}, \boldsymbol{j})$ of vertices with $\chi_{\boldsymbol{ij}} = 0$ and therefore have weight $z_{\boldsymbol{ij}} = 0$.

The following Theorem states that the problem of determining the SH inner product is equivalent to the problem of finding a maximum weighted clique.

Theorem 4.4 (Weighted clique correspondence theorem). *Let $X \Diamond_N Y$ be the inner product graph of attributed graphs $X, Y \in \mathcal{G}^n$. Then:*

1. *There is a bijective mapping*

$$\Phi^* : \mathcal{C}^*_{X\Diamond_N Y} \to \mathcal{I}(X, Y) \tag{15}$$

from the set $\mathcal{C}^*_{X\Diamond_N Y}$ of maximum weighted cliques to $\mathcal{I}(X, Y)$.

2. *There is a bijective mapping*

$$\Phi^+ : \mathcal{C}^+_{X\Diamond_N Y} \to \mathcal{S}_X \tag{16}$$

from the set $\mathcal{C}^+_{X\Diamond_N Y}$ of maximal cliques of $X \Diamond_N Y$ to the set $\mathcal{S}_X$ of all permutations acting on X.

3. *We have*

$$N^{-1}(\omega(C)) = \sigma(X, Y) \tag{17}$$

for all maximum weighted cliques $C \in \mathcal{C}^*_{X\Diamond_N Y}$ of $X \Diamond_N Y$.

4. *For each maximal clique $C \in \mathcal{C}^+_{X\Diamond_N Y}$ of $X\Diamond_N Y$ there is a permutation $\pi \in \mathcal{S}_X$ such that*

$$N^{-1}(\omega(C)) = \langle A(X^\pi), A(Y)\rangle. \tag{18}$$

Proof: Jain and Wysotzki (in press). □

From Theorem 4.4 follows that the inner product graph $X\Diamond_N Y$ is a compact representation of all possible inner products of $A(X^\pi)$ and $A(Y)$ where π passes through the whole set of permutations $\mathcal{S}_X$. Each permutation $\pi \in \mathcal{S}_X$ is uniquely encoded in the inner product graph $X\Diamond_N Y$ as a maximal clique C_π such that the inner product $\langle A(X^\pi), A(Y)\rangle$ can be determined from the weight $\omega(C_\pi)$ of the corresponding maximal clique. Moreover, the embeddings $\pi \in \mathcal{I}(X, Y)$ are in an one-to-one correspondence with the maximum weighted cliques.

A hopfield clique network

Let $Z \in \mathcal{W}^n$ be a weighted graph with adjacency matrix $A(Z) = (z_{ij})$. Without loss of generality we can assume that Z is normalized.[3]

The Hopfield clique network (HCN) $\mathcal{H}_Z$ associated with Z consists of n fully interconnected units. The synaptic weights w_{ij} between distinct units i and j are given by

$$w_{ij} = \begin{cases} w^+_{ij} \geq 0 & \text{if } (i, j) \in E(Z) \quad \text{(excitation)} \\ -w^-_{ij} < 0 & \text{if } (i, j) \notin E(Z) \quad \text{(inhibition)} \end{cases}$$

Thus Z uniquely determines the topology of $\mathcal{H}_Z$ (up to isomorphism). For this reason we identify the units of $\mathcal{H}_Z$ with the vertices of Z and excitatory connections between two units with edges between their corresponding vertices. Inhibitory connections in $\mathcal{H}_Z$ represent non-edges in Z.

The dynamical rule of $\mathcal{H}_Z$ is of the form

$$x_i(t+1) = x_i(t) + \sum_{j\neq i} w_{ij} o_j(t) + \theta_i \tag{19}$$

where $x_i(t)$ denotes the activation of unit i at time step t and θ_i is a constant external input applied to unit i. The output $o_i(t)$ of unit i is computed by a piecewise linear limiter transfer function of the form

$$o_i(t) = \begin{cases} 1 & x_i(t) \geq \tau_t \\ 0 & x_i(t) \leq 0 \\ x_i(t)/\tau_t & \text{otherwise} \end{cases} \tag{20}$$

where τ_t is a time dependent control parameter called *pseudo-temperature*. The output function $o_i(t)$ has a lower and an upper saturation point at 0 and 1, respectively. Starting with a sufficient large initial value τ_0 the pseudo-temperature is decreased according to an *annealing-schedule* to a final value τ_f.

The energy function of the network to be minimized is then of the form

$$E(t) = -\frac{1}{2}\sum_i \sum_{j\neq i} w_{ij} o_i(t) o_j(t) - \sum_i \theta_i o_i(t). \tag{21}$$

To simplify the following considerations we introduce some technical terms and notations:

1. *Excitatory and inhibitory degree*: Let $\deg_E(i)$ be the number of excitatory connections incident to unit i. We call

   $$\deg_E = \max\{\deg_E(i) \mid 1 \leq i \leq n\}$$

   the *excitatory degree* of $\mathcal{H}_Z$ and

   $$\deg_I = \max\{n - \deg_E(i) - 1 \mid 1 \leq i \leq n\}$$

   the *inhibitory degree* of $\mathcal{H}_Z$.
2. By $\theta^* = \max_i\{z_{ii}\}$ we denote the maximum vertex weight of Z.
3. Let $w^+, w^- > 0$ be positive constants bounded by

   $$w^+ < \frac{2}{n + \deg_I(\deg_E - 1)}$$

   $$w^- > \deg_E w^+ + \theta^*$$

Provided an appropriate parameter setting is given Theorem 4.5 states that the dynamical rule (19) performs a gradient descent with respect to the energy function E where the global (local) minima of E correspond to the maximum weighted (maximal) cliques of Z.

Theorem 4.5 (*Hopfield clique network theorem*). *Let Z be a normalized weighted graph and $\mathcal{H}_Z$ be a* HCN *associated with Z. Assume that $\tau_t \geq 1$ for all $t \geq 0$. If*

$$w_{ij}^+ = z_{ij} \cdot w^+ \tag{22}$$

$$w_{ij}^- = w^- \tag{23}$$

$$\theta_i = z_{ii} \cdot w^+ \tag{24}$$

for all $i, j \in V(Z)$, then:

1. *$E(t+1) \leq E(t)$ for all $t \geq 0$.*
2. *There is a bijective mapping between the local minima of E and the maximal cliques of Z.*
3. *There is a bijective mapping between the global minima of E and the maximum weighted cliques C of Z.*

Proof: Jain and Wysotzki (in press). □

From the first statement together with the fact that E is bounded follows that $\mathcal{H}_Z$ converges. Since the local minima of $E(t)$ correspond to maximal cliques, we can not guarantee that the network $\mathcal{H}_Z$ converges to an optimal solution corresponding to a maximum weighted clique. In addition the network can converge to unstable saddle points $\boldsymbol{u} \in \mathbb{R}^n$ of $E(t)$. Due to their instability, imposing random noise onto $\mathcal{H}_Z$ may shift the state vector $\boldsymbol{o}(t)$ away from $\boldsymbol{u}$. An example of an unstable saddle point is the trivial solution $\mathbf{0} \in \mathbb{R}^n$.

It is easy to verify that the following inequalities hold:

$$\begin{aligned} w_{ij}^{+} &< \frac{2}{n + \deg_I(\deg_E - 1)} \\ w_{ij}^{-} &> \deg_E w^{+} + \theta^{*} \end{aligned}$$

The upper bound of w_{ij}^{+} ensures that $\mathcal{H}_Z$ performs a gradient descent with respect to E. The lower bound of w_{ij}^{-} in guarantees that $\mathcal{H}_Z$ converges to an admissible solution provided that saddle points are avoided by imposing random noise onto the network.

Given a parameter setting satisfying the bounds of Theorem 4.5, then the HCN operates as follows: An initial activation is imposed on the network. Finding a maximum weighted clique then proceeds in accordance with the dynamical rule (19) until the system converges to a stable state. During evolution of the network any unit is excited by all active units with which it can form a clique and inhibits all other units. After convergence the stable state corresponds to a maximal clique C of Z. The units corresponding to the vertices of C can be identified by their respective non-negative activation.

4.2.2. Winner-takes all networks for maximum selection. This section describes *winner-takes-all* (WTA) networks as a neural network implementation of a maximum selector.

Let $\mathcal{V} = \{v_1, \ldots, v_K\}$ be a set consisting of distinct values $v_i \in \mathbb{R}$. One way to select the maximum from $\mathcal{V}$ within a connectionist framework are *winner-takes-all* (WTA) networks. The WTA net consists of K mutually inhibitory connected units. The activation $z_i(t)$ of unit c_i is initialized with v_i. After initialization the WTA network updates its state according to

$$z_i(t+1) = z_i(t) - w \cdot \sum_{j \neq i} [z_j(t)]_0 + I_i(t) \tag{25}$$

where $-w < 0$ represents the synaptic weight of the inhibitory connections and $I_i(t)$ is the external input of unit c_i. For $z \in \mathbb{R}$ the function

$$[z]_0 := \begin{cases} z & z > 0 \\ 0 & z \leq 0 \end{cases}$$

denotes the linear threshold function with lower saturation 0.

An *admissible solution* of the WTA network is a stable state where only one unit c_{i^*} has a positive activation $z_{i^*}(t) > 0$ and all other units $c_j \neq c_{i^*}$ are inhibited, i.e. $z_j(t) \leq 0$. The next result is due to Koutroumbas and Kalouptsidis (1994) and Lippman (1987). It shows, that under some assumptions, the WTA network converges to an admissible solution within finite time.

Theorem 4.6. *Let $\boldsymbol{I}(t) = \mathbf{0} \in \mathbb{R}^K$ and $\boldsymbol{z}(0) \in \mathbb{R}^K$ be the initial activation of the dynamical system* (25). *Suppose that $\boldsymbol{z}(0)$ has a unique maximum $z_{i^*(0)} > z_j(0)$ for all $j \neq i^*$. If*

$$0 < w < \frac{1}{K - 1}$$

then $\boldsymbol{z}(t)$ converges within finite time to an admissible solution.

For selecting the maximum of a given set $\mathcal{V}$ of input values it does not matter how the winner-takes-all character is implemented. In a practical setting one prefers a simple search algorithm for finding the maximum. However, a neural network implementation of the winner-takes-all selection principle is a fundamental prerequisite to accelerate decision making in classification and pattern recognition tasks of structures.

4.2.3. Extended winner-takes-all networks for attributed graphs. It is easiest to first introduce the architecture of X-WTA and then present X-WTA as an algorithm before describing the network implementation.

The* X-WTA *network. Let $X \in \mathcal{X}$ be a data graph, $\mathcal{Y} = \{Y_1, \ldots, Y_K\}$ be a set of model graphs.

A X-WTA network is a neural network consisting of two layers, a matching layer and an output layer. The matching layer has K subnets $\mathcal{H}_i = \mathcal{H}_{X \diamond Y_i}$, each of which is a HCN for matching an input X with model Y_i. The output layer is a competitive WTA network for maximum selection consisting of K inhibitory connected units c_i. Each subnet $\mathcal{H}_i$ is connected to output c_i.

The X-WTA network proceeds first by initializing the inhibition w of the WTA network and the subnets $\mathcal{H}_i$ with respect to the graphs X and Y_i of consideration. Once the X-WTA network has been initialized, it repeatedly executes three basic steps, *similarity matching*, *competition*, and *adaption* until one output unit wins the competition and all other output units are inhibited:

1. *Similarity Matching*. The `X-WTA` network simultaneously updates the current state vector $\boldsymbol{x}_i(t)$ of each subnet $\mathcal{H}_i$ according to the dynamical rule (19) giving $\boldsymbol{x}_i(t+1)$ as the new state of $\mathcal{H}_i$. Subsequently the `X-WTA` system computes *interim values* $\sigma_i(t+1)$ of the unknown `SH` inner products $\sigma(X, Y_i)$. An interim value $\sigma_i(t+1)$ is an approximation of $\sigma(X, Y_i)$ given the current state $\boldsymbol{x}_i(t+1)$ of $\mathcal{H}_i$ at time step $t+1$.
2. *Competition*. The interim values $\sigma_i(t)$ provide the basis for competition among the subnets via the `WTA` network. The classifier updates the current state $z(t)$ of the `WTA` network according to the dynamical rule (25) giving $z(t+1)$ as its new state vector. The external input $I_i(t)$ applied to unit c_i is of the form

   $$I_i(t) = I(\sigma_i(t))$$

   where I is a function transforming the interim values in a suitable form for further processing by the `WTA` network.
3. *Adaption*. The `X-WTA` network disables all subnets $\mathcal{H}_i$ for which $z_i(t+1) \leq 0$. Once a subnet is disabled it is excluded from the competition. A disabled subnet, which has not converged, aborts its computation and terminates before reaching a stable state.

Intertwined execution of the three basic steps, similarity matching, competition, and adaption implements the principle *elimination of competition*. This principle is based on the policy that a large `SH` inner product $\sigma(X, Y_i)$ emerges from a series of increasing large interim values $\sigma_i(t)$ at an early stage of the computation of $\mathcal{H}_i$. In detail the `X-WTA` network proceeds as follows:

Algorithm. Let X be a data graph and $\mathcal{Y} = \{Y_1, \ldots, Y_K\}$ a set of model graphs. Choose a value for the inhibition w satisfying

$$0 < w < \frac{1}{K-1}.$$

```
1. Enable all subnets $\mathcal{H}_i$. Each subnet can be enabled or disabled. Only if $\mathcal{H}_i$ is enabled it
   may proceed with its computation. Once $\mathcal{H}_i$ is disabled it resigns from the competition
   and is no longer in use.
2. Set $t = 0$.
3. Repeat
   (a) For each enabled subnet $\mathcal{H}_i$ do
         (i) Update $\mathcal{H}_i$ according to the dynamical rule (19) giving $\boldsymbol{x}_i(t+1)$.
        (ii) Compute an interim value $\sigma_i(t)$ of $\sigma(X, Y_i)$.[4]
       (iii) Compute the external input $I_i(t) = I(\sigma_i(t))$ of output unit $c_i$.
   (b) Update the WTA network according to the dynamical rule (25) giving $z(t+1)$.
   (c) Disable and terminate all subnets $\mathcal{H}_i$ for which the activation $z_i(t+1) \leq 0$.
   (d) Set $t = t + 1$.
   until $z_i(t) > 0$ for only one output unit $c_i$.
```

4. Let c_{i^*} be the winning unit. Assign X to the category represented by model Y_{i^*}.

Theorem 4.7 proves that, under some assumptions, the X-WTA classifier converges to an admissible solution after a finite number of iterations. An *admissible solution* refers to a uniquely defined winning unit of the X-WTA classifier algorithm.

Theorem 4.7. *Assume that the subnets $\mathcal{H}_1, \ldots, \mathcal{H}_K$ converge to a stable state within finite time. Then the* WTA *classifier converges after a finite number of iteration steps to an admissible solution.*

Proof: Jain and Wysotzki (in press). □

Example 4.1. Let Z_1, Z_2, and Z_3 be binary graphs. In this case the concepts of a maximum weighted clique and a maximum clique are equivalent. By $\mathcal{H}_i$ we denote the HCN associated with Z_i for $i = 1, 2, 3$. Figure 3 depicts a functional diagram of the X-WTA network for $K = 3$. Excitatory connections are represented by solid lines and inhibitory connections by dotted lines. The subnets $\mathcal{H}_1$, $\mathcal{H}_2$, and $\mathcal{H}_3$ in the matching layer are excitatory connected to the output units c_1, c_2, and c_3 via on-off switches. The switches enable or disable a subnet for or from further computation. The subnets $\mathcal{H}_1$ and $\mathcal{H}_3$ are enabled, while $\mathcal{H}_2$ is disabled. The shading of an output unit c_i indicates its activation $z_i(t)$. Brighter shading corresponds to lower activation and vice versa. A white shading represents output units c_i with activation $z_i(t) \leq 0$. Thus unit c_1 has the largest activation and unit c_2 the lowest. The white shading of unit c_2 indicates that $z_2(t) \leq 0$. In this case subnet $\mathcal{H}_2$ is disabled as illustrated by the off-position of the corresponding switch. In contrast the switches connecting $\mathcal{H}_1$ and $\mathcal{H}_3$

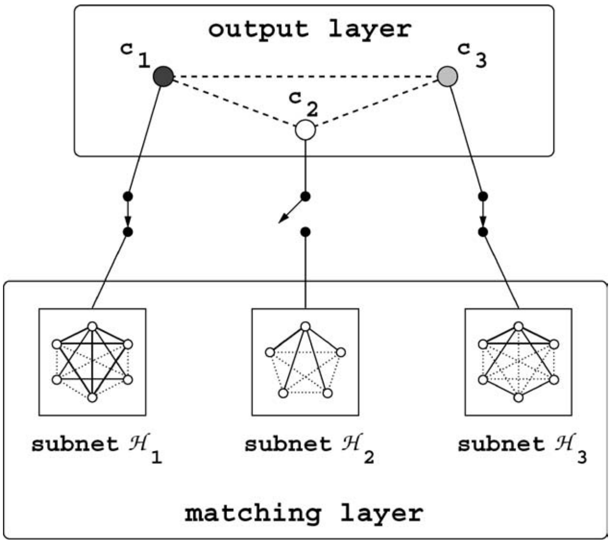


Figure 3. Architecture of a WTA classifier.

with c_1 and c_3, resp., are in an on-position. Thus the subnets $\mathcal{H}_1$ and $\mathcal{H}_3$ are enabled and further participate in the competition. This instantaneous state of the X-WTA network is consistent with the results we would expect if the subnets $\mathcal{H}_i$ converge to optimal solutions corresponding to maximum cliques. For each subnet a maximum clique of the associated graph is highlighted.

In the remainder of this section we briefly discuss some issues of the X-WTA network algorithm. For a more detailed presentation we refer to Jain and Wysotzki (in press).

Interim values. To motivate interim values consider that decision making can be regarded as a process of sufficiently reducing uncertainty and doubt about alternatives to allow a reasonable choice to be made from among them. In this context the interim values of all subnets $\mathcal{H}_i$ model an information gathering function of decision making to reduce uncertainty. Progressing evolution of the subnets $\mathcal{H}_i$ provide interim values, which aim at reducing that uncertainty.

The Role of inhibition. The choice of the inhibitory weight $-w$ in the WTA network is essential for the gain in speed without substantial loss of classification accuracy with respect to the canonical MS classifier. A large value for w will tend to fast convergence and hence give a large error rate. We can decrease the error rate by decreasing w, but if this is taken too far, the convergence time of a X-WTA classifier approaches the convergence time of a MS classifier, which may be intractable in a practical setting. This trade-off between speed and classification accuracy is crucial for applying a X-WTA classifier to practical problems.

Synchronization of the subnets. The number of time steps until a HCN $\mathcal{H}$ converges to a stable state depends on the number of units of $\mathcal{H}$. Different sizes and relaxation times of the subnets of the X-WTA classifier may may bias decision making. One way around this problem is to synchronize the relaxation of the subnets. Slower subnets may update their state more often than faster subnets before performing the next update step of the units in the output layer.

4.3. *A* KMS *algorithm using* X-WTA *networks*

This section assembles the results of the previous sections to formulate a K-means algorithm for structures by means of a X-WTA network and concludes with a discussion about the limitations and benefits of the proposed neural KMS algorithm.

Let $\mathcal{X} = \{X_1, \ldots, X_M\}$ be a set of data graphs. First initialize a set $\mathcal{Y} = \{Y_1, \ldots, Y_K\}$ of model graphs. After initialization of $\mathcal{Y}$ repeat the following steps until termination:

1. To classify all samples $X \in \mathcal{X}$ apply each X to the X-WTA network. Applying X to X-WTA involves construction of K inner product graphs $Z_i = X \diamondsuit Y_i$ to map the task of computing $\sigma(X, Y_i)$ to the task of finding a maximum weighted clique C_i with $\omega(C_i) = \sigma(X, Y_i)$. To find a maximum weighted clique in Z_i transform the inner product graphs Z_i to subnets $\mathcal{H}_i$.

After initialization let the X-WTA network evolve until convergence to select a winning model Y_{i^*}. Assign X to the cluster represented by Y_{i^*}.

2. After each sample graph has been classified recompute the incremental means of the current clusters according to Step (2c) of figure 2. Suppose that $X_{i_1}, \ldots, X_{i_{n_j}}$ have been assigned to model Y_j by the X-WTA network in the previous step. Initially set $Y_{j,1} = X_{i_1}$. Then iteratively embed the samples X_{i_k} to the current incremental mean $Y_{j,k-1}$ to obtain

$$Y_{j,k} = \frac{k-1}{k} Y_{j,k-1} + \frac{1}{k} X_{i_k}^{\pi_{i_k}}$$

where $\pi_{i_k} \in \mathcal{I}(X_{i_k}, Y_{j,k-1})$ is an embedding. To find an embedding, construct an inner product graph $Z_{i_k,j} = X_{i_k} \diamond Y_{j,k-1}$ and transform the inner product graph $Z_{i_k,j}$ to a Hopfield Clique network $\mathcal{H}_{i_k,j}$. The task of the Hopfield network is to find a maximum weighted clique of $Z_{i_k,j}$ corresponding to an embedding from X_{i_k} to $Y_{j,k-1}$. According to Theorem 4.5 the clique C found by $\mathcal{H}_{i_k,j}$ is a maximal clique and in the ideal case a maximum weighted clique. By (15) and (16) of Theorem 4.4 the maximal clique C corresponds to a permutation $\pi_C \in \mathcal{S}_n$ which is an embedding from X_{i_k} to $Y_{j,k-1}$ if and only C is a maximum weighted clique.

Figure 4 summarizes the KMS algorithm.

Limitations and benefits. Besides the well known limitations of the K-means algorithms in the domain of feature vectors, additional problems arise when dealing with attributed graphs. From a theoretical point of view, the main drawback of central clustering in the domain of graphs is that the notion of a sample mean is not uniquely determined. This makes an analysis of convergence properties difficult and may result in an instable or oscillating behavior of central clustering algorithms like KMS.

From the point of view of computational complexity, determining a sample mean of a set of data graphs is in fact futile even for small sets of data graphs due to the high combinatorial explosion. Thus any practical applicable central clustering algorithm will resort to approximations of the sample mean at the expense of finding a global minimum of the cost function defined in (4). In addition, solving a large number of NP-complete graph matching problems to identify the optimal reference model of a given input graph and to find an embedding for recomputing the means may hinder the practical use of central clustering of graphs. The same way out of this problem is to use heuristics which find suboptimal solutions within an acceptable time scale. Provided the given data set carries an explicit cluster structure, then the general assumption of using approximations of the sample mean and structural distortion values is that elements of the same clusters share large parts of similar substructures which will be largely identified even by suboptimal approaches.

Nevertheless, even approximately solving the graph matching problem is a computational intensive procedure which appears to be intractable for practical graph clustering problems. The X-WTA mechanism is a first step to reduce the number of graph matching problems to be actually solved. Given an input graph, hard optimizers like the KMS algorithm require the explicit computation (or approximation) of only one instead of K structural distortions.

```
KMS-Algorithm:

  – $\mathcal{X} = \{X_1, \dots, X_N\} \subseteq \mathcal{G}^n$ be a set of data graphs.
  – $M = (m_{ij}) \in \mathcal{M}_{N \times K}(\{0,1\})$ be the membership matrix.

1. Initialize $K$ and $\mathcal{Y} = \{Y_1, \dots Y_K\}$.
2. Repeat
   a) Set $m_{ij} = 0$ for all $i, j$
   b) For all $i = 1, \dots, N$:
      i) Apply $X_i$ to the X-WTA network.
      ii) Set $m_{ii^*} = 1$ if $Y_{i^*}$ wins the competition.
   c) For all $j = 1, \dots, K$:
      i) Let $\{i \mid m_{ij} = 1\} = \{i_1, \dots, i_{n_j}\}$.
      ii) Set $Y_j = X_{i_1}$
      iii) For all $k = 2, \dots, n_j$:
         – Compute $\pi_{i_k} \in \mathcal{I}(X_{i_k}, Y_{j,k-1})$ with a HCN.
         – Compute
              $Y_{j,k} = \frac{k-1}{k} Y_{j,k-1} + \frac{1}{k} X_{i_k}^{\pi_{i_k}}$
3. until no noticeable changes in $\mathcal{Y}$ are observed.
4. Return cluster centers $\mathcal{Y} = \{Y_1, \dots Y_K\}$.
```

Figure 4. Outline of the KMS algorithm using X-WTA networks.

One limitation of the X-WTA network is its restricted use to soft optimizers. When applied to soft optimizers like self-organizing feature maps, fuzzy clustering or stochastic clustering algorithms derived from the maximum entropy principle, the gain in computational time becomes less or even is lost. This is due to the fact that in general soft optimizers adjust more than one model for a given input graph and therefore require the explicit calculation of more than one structural distortion.

A second limitation of the X-WTA network is that its classification accuracy is poor if the models are pairwise similar. This requires a more careful initialization of the models than in clustering methods which rely on the nearest neighbor rule. In addition grouping a set of similar data graphs with an explicit cluster structure fails when using an X-WTA network to assign the data graphs to their respective optimal reference models. To illustrate this problem consider as an extreme case a data set consisting of several isomorphic copies of K similar model graphs. This data set carries a trivial cluster structure which might not be extracted by the proposed neural KMS algorithm.

From the above discussion follows that an X-WTA network is best suited for hard optimizers to solve central clustering problems of data sets of attributed graphs, where element belonging to the same cluster are similar and elements from different clusters are significantly dissimilar.

It is left to discuss the use of Hopfield networks. Although Hopfield networks have been applied with varied success to approximately solve combinatorial approximation problems, they possess some attractive features which justify their use in the context of central clustering: (1) As any neural network, Hopfield networks are error tolerant and robust against noise. Error tolerance is required when dealing with graphs where structural errors occur due to missing vertices or edges. Robustness against noise is necessary to cope with noisy vertex and edge attributes. (2) There is a solid theoretical foundation, in particular for solving the weighted maximum clique problem. (3) It is quite natural and straightforward to imbed Hopfield network into an X-WTA scheme. It is in fact the third feature which is the crucial factor to apply Hopfield network for Step (2b) of the neural KMS algorithm.

The particular choice of heuristic to approximately solve graph matching algorithms to determine the embeddings in Step (2c) is problem dependent. For sake of simplicity and convenience of presentation we stick to Hopfield networks.

5. Experiments

The aim of this section is to demonstrate that central clustering methods like the neural KMS algorithm can be applied to graph clustering problems despite the elusiveness of formulating analytical concepts in the domain of graphs. The experiments are intended not so much as a comprehensive study but rather as an illustration of the specific properties of the KMS algorithm. As a consequence results are presented with an emphasis on conceptual issues rather than experimental exhaustion. In Section 5.1 we apply the KMS algorithm to synthetical data and in Section 5.2 to images of arm postures.

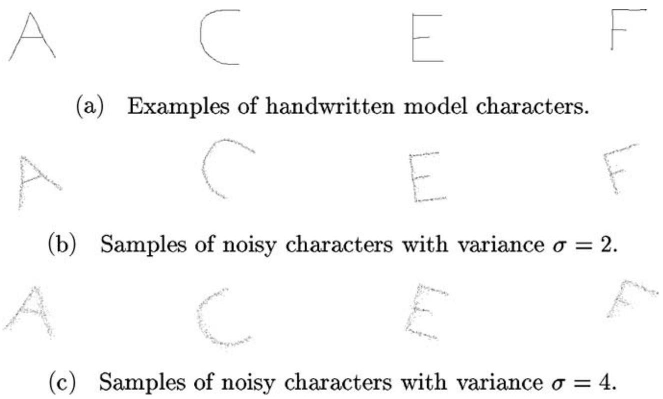
All algorithms were implemented in Java using JDK 1.2. The experiments were run on a multi-server Sparc SUNW Ultra-4.

5.1. Synthetic characters

The aim of our first experiment is to show that central clustering of attributed graphs is capable to simultaneously perform an inner clustering of the attribute vectors and an outer clustering of the structural variation of the graphs.

We used synthetic data to emulate clustering of handwritten characters as they typically occur in pen technology of small hand-held devices, for example PDAs. We do not apply additional on-line information. First we have drawn all 26 characters $\{\mathtt{A}, \dots, \mathtt{Z}\}$ using an X windows interface. The contours of each image were discretized and expressed as a set of points in the 2D plane.

For each character we generated $N = 10$ corrupted data characters as follows: First we randomly rotated the model image by an angle $\alpha_k \in [-37^\circ, +37^\circ]$. Then to each point we added $N(0, \sigma)$ Gaussian noise with standard deviation $\sigma = 2$. Each point had 10% probability to be deleted. From each point set we randomly selected points such that the pairwise normalized distances between the chosen points is larger than a given threshold θ. We transformed this point set $\mathcal{P}$ to a fully connected attributed graph. The vertices $v(\boldsymbol{p})$ represent the points $\boldsymbol{p} \in \mathcal{P}$ and the edges represent an abstract line between two points. To each vertex $v(\boldsymbol{p})$ we assigned a three dimensional attribute vector $\boldsymbol{a} = (a_1, a_2, a_3)$. The



(a) Examples of handwritten model characters.

(b) Samples of noisy characters with variance $\sigma = 2$.

(c) Samples of noisy characters with variance $\sigma = 4$.

Figure 5. Examples of model and corrupted data characters.

first attribute a_1 is the normalized distance of the point $\boldsymbol{p}$ to the center of gravity of the corrupted image. The second attribute a_2 is the mean distance of the normalized distances from point $\boldsymbol{p}$ to all other points $\boldsymbol{q} \in \mathcal{P}$. Finally, the third attribute a_3 is the variance of the normalized distances between $\boldsymbol{p}$ an all other points $\boldsymbol{q} \in \mathcal{P}$. To each edge connecting vertex $v(\boldsymbol{p})$ and $v(\boldsymbol{q})$ we assigned a two dimensional attribute vector $\boldsymbol{b} = (b_1, b_2)$. The first attribute is the normalized distance between $\boldsymbol{p}$ and $\boldsymbol{q}$. The second attribute measures the normalized distance between the center of gravity of the corrupted image and the abstract line passing through $\boldsymbol{p}$ and $\boldsymbol{q}$. Thus each graph is a representation of a character, which is invariant to rotation, translation, and scaling.

Given the dataset consisting of 260 corrupted samples we computed the pairwise distances of the samples to identify a subset of characters for which their respective samples form a cluster. Figure 6 visualizes the dissimilarity matrix where the samples are lexicographically ordered. Dark gray levels correspond to high dissimilarity. From figure 6 we see that there are only few compact clusters of samples representing the same character. Based on this distribution we selected the *well-behaved* characters $\mathcal{Y} = \{\mathtt{A}, \mathtt{C}, \mathtt{E}, \mathtt{F}, \mathtt{I}, \mathtt{L}, \mathtt{V}\}$. The model images of the first four characters are shown in figure 5(a).

We created two datasets, each consisting of 70 sample graphs. For each dataset and each of the $K = 7$ chosen model characters $Y \in \mathcal{Y}$ we generated $N = 10$ corrupted images and transformed these images to attributed graphs following the same scheme as described for the whole alphabet. The first dataset $\mathcal{D}_2$ was sampled by using Gaussian noise with standard deviation $\sigma = 2$. The second dataset $\mathcal{D}_4$ was created by imposing Gaussian noise with standard deviation $\sigma = 4$. Figure 5(b) and (c) show examples of corrupted images of both datasets.

Table 1 shows the structural variation of the graphs of dataset $\mathcal{D}_2$ and $\mathcal{D}_4$, respectively. For both tables, the first row specifies the mean, the second row the variance, the third row the minimal, and the fourth row the maximal number of vertices of the sample graphs representing the character specified by the column identifier.

We applied the neural `KMS` algorithm to both datasets. For dataset $\mathcal{D}_2$ the algorithm converged after two and for D_4 after six epochs into a local minimums of the average

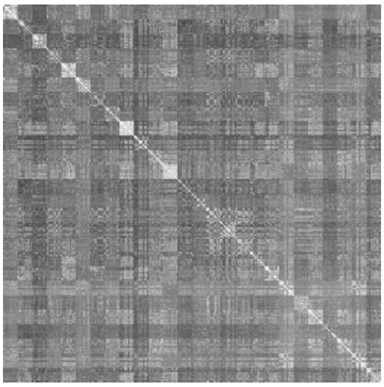


Figure 6. $2D$ visualizations of pairwise dissimilarities between all 260 attributed graphs representing corrupted images of the alphabet. Dark gray levels correspond to high dissimilarity.

distortion $E(M, \mathcal{Y}, \mathcal{X})$. The convergence time of KMS applied on $\mathcal{D}_2$ was about 20 seconds and on $\mathcal{D}_4$ about 3 minutes. The results indicate that more difficult clustering problems require much more time even if the order of the sample graphs is much smaller. In contrast, the approach proposed by Gold, Rangarajan, and Mjolsness (1996) applied on 64 weighted graphs of order 10 and 4 cluster centers required about 10 minutes until convergence. Unfortunately, almost no experimental work on graph clustering using graph matching methods addressed the issue of computational time required for grouping the data.

Table 2 shows the confusion matrices after clustering the samples of $\mathcal{D}_2$ and $\mathcal{D}_4$, respectively. Each row shows the distribution of the samples of a characters into the seven clusters. According to the nearest neighbor rule, the models obtained by the clustering procedure misclassified four samples of dataset $\mathcal{D}_2$ and only three samples of $\mathcal{D}_4$. The results show that the neural KMS algorithm is able to simultaneously extract an inner statistical cluster structure of the vertex and edges attribute vectors and an outer structural cluster structure of the graphs.

Table 1. Structural variations of sample graphs of dataset $\mathcal{D}_2$ (upper part) and $\mathcal{D}_4$ (lower part).

	A	C	E	F	I	L	V
$\mathcal{D}_2$							
μ	44.4	70.2	42.6	48.2	44.5	39.3	53.6
σ^2	1.8	10.4	4.8	11.4	5.3	4.2	6.4
min	42.0	64.0	39.0	45.0	41.0	35.0	50.0
max	46.0	75.0	46.0	57.0	47.0	42.0	59.0
$\mathcal{D}_4$							
μ	24.4	37.5	25.2	27.7	25.8	24.8	30.5
σ^2	2.6	2.5	2.2	3.0	5.2	5.0	2.5
min	22.0	35.0	23.0	24.0	22.0	21.0	28.0
max	27.0	40.0	28.0	31.0	29.0	29.0	34.0

Table 2. Confusion matrices. Each row shows the distribution of all samples of a character into clusters. The upper part refers to the dataset $\mathcal{D}_2$ and the lower part to the dataset $\mathcal{D}_4$.

	C_A	C_C	C_E	C_F	C_I	C_L	C_V
$\mathcal{D}_2$							
A	10	0	0	0	0	0	0
C	0	10	0	0	0	0	0
E	0	0	9	0	1	0	0
F	0	0	0	9	0	0	1
I	1	0	0	0	9	0	0
L	1	0	0	0	0	9	0
V	0	0	0	0	0	0	10
$\mathcal{D}_4$							
A	10	0	0	0	0	0	0
C	0	10	0	0	0	0	0
E	0	0	9	0	0	0	1
F	0	0	0	10	0	0	0
I	0	0	0	0	8	1	1
L	0	0	0	0	0	10	0
V	0	0	0	0	0	0	10

5.2. Sensing people

One fundamental problem in computer vision and human-machine interaction is to sense gestures and postures of peoples for directing computers or robots. This problem arises in many areas like, for example, policemen giving signs to regulate the traffic, directing a crane, guiding a vehicle into a parking lot, or medical monitoring of patients in a hospital or nursing home.

In this experiment we apply the KMS algorithm to cluster arm postures of a person as shown in figure 7. Five different classes of 235 postures are considered: (P_0) Unknown, (P_1) NoArms, (P_2) RightArm, (P_3) LeftArm, and (P_4) BothArms, each referring to the lifted arms of a person. Table 3 gives a concise class description where 0 denotes an arm

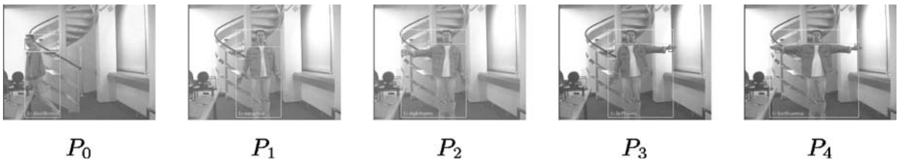


Figure 7. Sample images of arm postures.

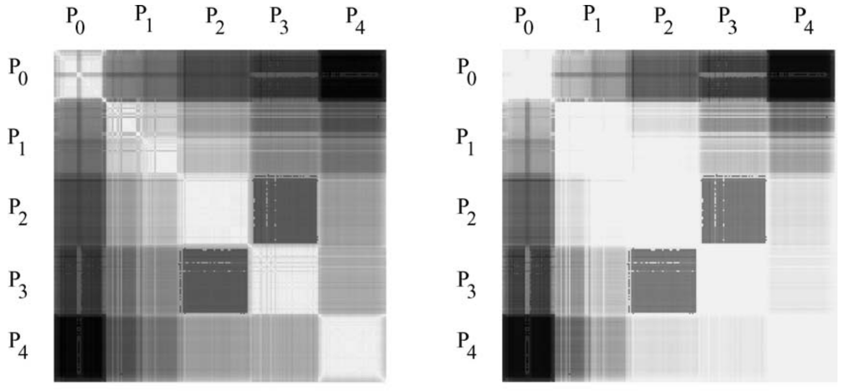
Table 3. Class description of arm postures.

Class	Right arm	Left arm	Class distribution
P_0	$*$	$*$	37
P_1	0	0	50
P_2	1	0	50
P_3	0	1	50
P_4	1	1	48

pointing downwards and 1 denotes an arm pointing straight out from the shoulders parallel to the floor. By $*$ we denote the *don't care* symbol.

Each image was obtained by automatically localizing a person in video data from a camera. The localized person is enclosed in a bounding box. Position of head and both hands of the person are identified by skin color. We transformed each image to a fully connected attributed graph. The vertices represent upper corners of the bounding box and the identified body parts. To each vertex we attach a three dimensional binary attribute vector $\boldsymbol{a} = (a_1, a_2, a_3) \in \{0, 1\}^3$. Attribute a_1 refers to the left corner of the bounding box, a_2 to the right corner, and a_3 to one body part, that is to the head or one hand. The value 1 of an attribute a_i denotes the presence and the value 0 the absence of the i-th feature. Edges between two vertices are weighted by the distance between the corresponding components in the image.

Figure 8 shows two different presentations of the matrix of pairwise Euclidean distances between the different attributed graphs. Dark gray levels correspond to high distance values. The patterns are ordered with respect to their classes. From figure 8(a) we that the patterns resolve into 5 clusters each corresponding to one of the 5 postures. Patterns of class P_1, P_2,



(a) Normalized gray levels. (b) Image after enhancing contrast.

Figure 8. Different presentations of $2D$ visualizations of pairwise dissimilarities between all 235 arm postures. Dark gray levels correspond to high dissimilarity.

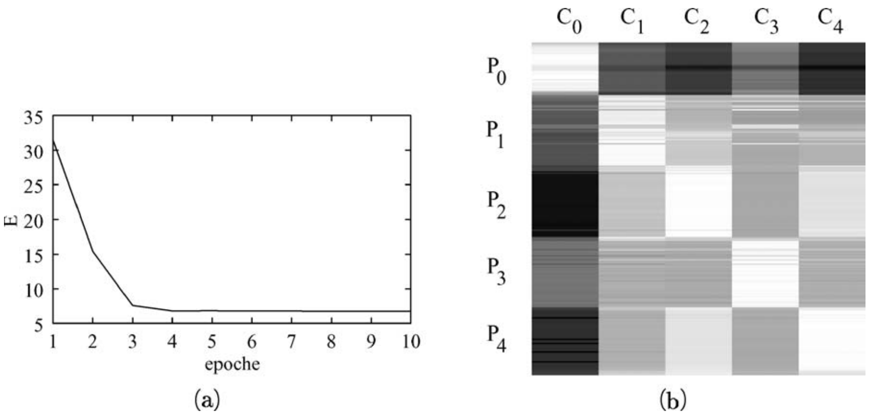


Figure 9. Results of clustering arm postures: (a) Average distortion as a function of the number of epochs. (b) Distance matrix between models representing the clusters $C_1, \ldots, C_4$ and the input samples ordered by arm posture $P_0, \ldots, P_4$.

patterns of class P_2, P_4 and patterns of class P_3, P_4 do not exhibit a clearly marked cluster structure. This is best seen when enhancing contrast as shown in figure 8(b).

We randomly selected five patterns from the dataset to initialize the models. After seven epochs the average distortion $E(M, \mathcal{Y}, \mathcal{X})$ has converged to a local minimum. Figure 9(a) shows the average distortion as a function of the number of epochs. A small increase of the cost function occured after the fifth epoch. This effect is due to the approximative nature of the proposed algorithm and the non-uniqueness of the incremental mean.

The quality of the cluster structure discovered by KMS is visualized in figure 9(b). From that figure we see that the cluster structure of the data set was recognized. The cluster centers of $C_1, \ldots, C_4$, however, are pairwise more similar than expected from the visualization of the pairwise distance values in figure 8(a).

Table 4 shows the confusion matrices before and after clustering. The ith row of both confusion matrices shows the distribution of members of class P_i into clusters $C_1, \ldots, C_4$. In terms of classification accuracy using the nearest neighbor rule, the randomly chosen initial models misclassified about 30% and the models obtained by the clustering procedure misclassified about 8% of the samples.

Table 4. Confusion matrices. Each row shows the distribution of all samples of a posture P_i into clusters. Left: Before clustering. Right: After clustering.

	C_0	C_1	C_2	C_3	C_4	C_0	C_1	C_2	C_3	C_4
P_0	30	5	2	0	0	35	0	0	2	0
P_1	0	25	25	0	0	0	43	0	7	0
P_2	0	0	50	0	0	0	0	49	0	1
P_3	0	0	0	12	38	0	3	0	45	2
P_4	0	0	0	0	48	0	0	5	0	43

6. Conclusion

The first part of this paper addressed the issue of missing analytical methods in the domain of attributed graphs. We presented a mathematical framework of metric graph spaces to place central clustering of attributed graphs on a principled footing. The metric graph space is equipped with an Euclidean distance of graphs induced by the Schur-Hadamard inner product. The Euclidean distance gives rise to introduce concepts as weighted mean and sample mean of a set of data graphs, which constitute the basis of central clustering in the domain of graphs. As an example how to utilize the properties of a metric graph space for central clustering we formulated a generic K-means algorithm for structures.

The second addressed the issue of computational complexity of central clustering algorithms applied on graphs. We proposed a neural network solution for the generic KMS algorithm. The distinguishing feature of the proposed algorithm is the use of an X-WTA network which assigns a given input graph to its reference model without explicitly computing structural distortions. The underlying principle of this approach is *elimination of competition* where the models constitute the competitors and the winning model of the competition is considered as the optimal reference model of the current input graph. Following this approach, the overall gain in computational time is reduced from K to one complete calculation of structural distortions between the models and the current input graph. This hypothesis is supported by our preliminary experiments. Moreover, the experiments show that central clustering of graphs simultaneously performs an inner clustering of the attributes and an outer structural clustering of the structural variations of the graphs.

In forthcoming papers we use the Schur-Hadamard inner product to extend supervised neural learning machines from the domain of feature vectors to the domain of graphs.

Acknowledgments

We would like to thank Stefan Bischoff from the Heinrich Hertz Institute, Berlin, for providing us the arm posture image data.

Notes

[1] The median graph as defined by Bunke et al. also intuitively complies to a sample mean of a set of graphs.

[2] An inner product is a positive definite, symmetric bilinear form $\langle \, , \rangle : \mathcal{V} \times \mathcal{V} \rightarrow \mathbb{R}$.

[3] If Z is not normalized, we can first transform Z to a normalized graph Z', compute the solution in Z', and back-transform the solution to Z.

[4] For convenience of presentation the interim values $\sigma_i(t)$ are computed with respect to $\boldsymbol{x}_i(t)$ rather than $\boldsymbol{x}_i(t+1)$.

References

Ballard, D. H., & Brown, C. M. (1982). *Computer Vision*. Prentice Hall, Englewood Cliffs, New Jersey.

Ballard, D. H., Gardner, P. C., & Srinivas, M. A. (1987). Graph problems and connectionist architectures. Technical Report TR 167, Dept. Computer Science, University of Rochester.

Bezdek, J. C. (1981). *Pattern recogniton with fuzzy objective function algorithms*. New York, NY: Plenum Press.

Bunke, H., Foggia, P., Guidobaldi, C., & Vento, M. (2003). Graph clustering using the weighted minimum common supergraph. In E. Hancock and M. Vento (Eds.), *Graph based representations in pattern recognition. 4th IAPR international workshop, GbRPR 2003*, LNCS 2726, Springer-Verlag (pp. 235–246).

Bunke, H., Münger, A., & Jiang, X. (1999). Combinatorial search versus genetic algorithms: A case study based on the generalized mean graph problem. *Pattern Recognition Letters*, *20:11–13*, 1271–1277.

Carpenter, G. A., & Grossberg, S. (1987a). A massively parallel architecture for a self-organising neural pattern recognition machine. *Computer Vision, Graphics, and Image Processing, 37*, 54–115.

Carpenter, G. A., & Grossberg, S. (1987b). ART 2: Self-organisation of stable category recognition codes for analog input patterns. *Applied Optics, 26*, 4919–4930.

Carpenter, G. A., & Grossberg, S. (1988). The ART of adaptive pattern recognition by a self-organizing neural network. *Computer, 21:3*, 77–88.

Christmas, W. J., Kittler, J., & Petrou, M. (1995). Structural matching in computer vision using probabilistic relaxation. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 17:8*, 749–764.

Cross, A. D. J., Wilson, R. C., & Hancock, E. R. (1997). Inexact graph matching with genetic search. *Pattern Recognition, 30:6*, 953–970.

Eshera, M. A., & Fu, K. S. (1986). An image understanding system using attributed symbolic representation and inexact graph-matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 8:5*, 604–618.

Garey, M., & Johnson, D. (1979). *Computers and intractability: A guide to the theory of NP-completeness*. W.H. Freeman and Company, New York.

Gold, S., & Rangarajan, A. (1996). A graduated assignment algorithm for graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 18:4*, 377–388.

Gold, S., Rangarajan, A., & Mjolsness, E. (1996). Learning with preknowledge: Clustering with point and graph matching distance measures. *Neural Computation, 8:4*, 787–804.

Goldfarb, L., Abela, J., Bhavsar, V. C., & Kamat, V. N. (1995). Can a vector space based learning model discover inductive class generalization in a symbolic environment? *Pattern Recognition Letters, 16*, 719–726.

Günter, S., & Bunke, H. (2002). Adaptive self-organizing map in the graph domain. In H. Bunke and A. Kandel (Eds.), *Hybrid methods in pattern recognition*, World Scientific (pp. 61–74).

Günter, S., & Bunke, H. (2002). Self-organizing map for clustering in the graph domain. *Pattern Recognition Letters, 23*, 401–417.

Hagenbuchner, M., Sperduti, A., & Tsoi, A. C. (2003). A self-organizing map for adaptive processing of structured data. *IEEE Transactions on Neural Networks, 14:3*, 491–505.

Herault, L., Horaud, R., Veillon, F., & Niez, J. J. (1990). Symbolic image matching by simulated annealing. In *Proceedings of British Machine Vision Conference* (pp. 319–324).

Hofmann, T., & Buhmann, J. M. (1997). Pairwise data clustering by deterministic annealing. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 19:1*, 1–14.

Hopfield, J. J., & Tank, D. W. (1985). Neural computation of decisions in optimization problems. *Biological Cybernetics, 52*, 141–152.

Palmer, R. G., Hertz, J., & Krogh, A. (1991). *Introduction to the theory of neural computation*. Redwood City, CA: Addison-Wesley.

Jain, B. J., & Wysotzki, F. The maximum ω-clique problem and the Hopfield ω-clique model. Submitted for publication.

Jain, B. J., & Wysotzki, F. The Schur-Hadamard inner product and maximum ω-cliques in an inner product graph. Submitted for publication.

Jain, B. J., & Wysotzki, F. Winner-takes-all classification of structures. Submitted for publication.

Jain, B. J., & Wysotzki, F. (2001). Distance-based classification of structures within a connectionist framework. In R. Klinkenberg et al. (Eds.), *Proceedings Fachgruppentreffen Maschinelles Lernen*.

Jain, B. J., & Wysotzki, F. (2002). Self-organizing recognition and classification of relational structures. In W. Gray and C. Schunn (Eds.), *The 24th Annual meeting of the cognitive science society*. Springer Verlag (pp. 163–173).

Jain, B. J., & Wysotzki, F. (2003). A competitive winner-takes-all architecture for classification and pattern recpgnition of structures. In E. Hancock and M. Vento (Eds.), *Graph based representations in pattern recognition. 4th IAPR international workshop, GbRPR 2003*, LNCS 2726, Springer-Verlag (pp. 259–270).

Jain, B. J., & Wysotzki, F. (2003). A k-winner-takes-all classifier for structured data. In A. Günter, R. Kruse, and B. Neumann (Eds.), *KI 2003: Advances in artificial intelligence. 26th annual conference on AI*, LNAI 2821, Springer-Verlag (pp. 342–354).

Jain, B. J., & Wysotzki, F. (2003). Perceptron learning in the domain of graphs. In *Proc. of the International Joint Conference on Neural Networks, IJCNN 2003*.

Jiang, X., Münger, A., & Bunke, H. (1999). Synthesis of representative symbols by computing generalized median graph. In *Proceedings of the 3rd International Workshop on Graphics Recognition, GREC'99, Jaipur* (pp. 187–194).

Jiang, X., Münger, A., & Bunke, H. (2001). On median graphs: Properties, algorithms, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 23:10*, 1144–1151.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics, 43*, 59–69.

Koutroumbas, K., & Kalouptsidis, N. (1994). Qualitative analysis of the parallel and asynchronous modes of the hamming network. *IEEE Transactions on Neural Networks, 15:3*, 380–391.

Lippman, R. P. (1987). An introduction to computing with neural nets. *IEEE ASSP Magazine* (pp. 4–22).

Lozano, M. A., & Escolano, F. (2003a). ACM attributed graph clustering for learning classes of images. In E. Hancock and M. Vento (Eds.), *Graph based representations in pattern recognition. 4th IAPR international workshop, GbRPR 2003*, LNCS 2726, Springer-Verlag (pp. 247–258).

Lozano, M. A., & Escolano, F. (2003b). Em algorithm for clustering an ensemble of graphs with comb matching. In A. Rangarajan, M. Figueiredo, and J. Zerubia (Eds.), *Energy minimization methods in computer vision and pattern recognition, EMMCVPR 2003*, LNCS 2683, Springer-Verlag (pp. 52–67).

Luo, B., Robles-Kelly, A., Torsello, A., Wilson, R. C., & Hancock, E. R. (2001a). Discovering shape categories by clustering shock trees. In W. Skarbek (Ed.), *Computer analysis of images and patterns: 9th international conference, CAIP 2001*, LNCS 2124, Springer-Verlag (pp. 152–160).

Luo, B., Robles-Kelly, A., Torsello, A., Wilson, R. C., & Hancock, E. R. (2001b). A probabilistic framework for graph clustering. In *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 912–919).

Luo, B., Wilson, R. C., & Hancock, E. R. (2003). Spectral embedding of graphs. *Pattern Recognition, 36:10*, 2213–2223.

Pelillo, M., Siddiqi, K., & Zucker, S. W. (1999). Matching hierarchical structures using association graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 21:11*, 1105–1120.

Rangarajan, A., Gold, S., & Mjolsness, E. (1996). A novel optimizing network architecture with applications. *Neural Computation, 8:5*, 1041–1060.

Ruspini, E. H. (1969). A new approach to clustering. *Information Control, 15*, 22–32.

Sanfeliu, A., Serratosa, F., & Alquezar, R. (2000). Clustering of attributed graphs and unsupervised synthesis of function-described graphs. In *Proceedings of the 15th International Conference on Pattern Recognition, ICPR'2000, Barcelona, Spain* (pp. 1026–1029).

Sanfeliu, A., Serratosa, F., & Alquezar, R. (2002). Synthesis of function-described graphs and clustering of attributed graphs. *International Journal of Pattern Recognition and Artificial Intelligence, 16:6*, 621–655.

Schädler, K., & Wysotzki, F. (1999). Comparing structures using a Hopfield-style neural network. *Applied Intelligence, 11*, 15–30.

Sengupta, K., & Boyer, K. (1995). Organising large structural databases. *IEEE Transaction on Pattern Analysis and Machine Intelligence, 17:4*, 321–332.

Shapiro, L. G., & Haralick, R. M. (1981). Structural descriptions and inexact matching. *IEEE Transaction on Pattern Analysis and Machine Intelligence, 3*, 514–519.

Suganthan, P., Teoh, E., & Mital, D. (1995). Pattern recognition by graph matching using potts mft networks. *Pattern Recognition, 28*, 997–1009.

Williams, M. L., Wilson, R. C., & Hancock, E. R. (1999). Deterministic search for relational graph matching. *Pattern Recognition, 32:7*, 1255–1271.

Wong, A., & You, M. (1985). Entropy and distance of random graphs with application to structural pattern recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 7:5*, 599–609.

Yu, S.-S., & Tsai, W.-H. (1992). Relaxation by the Hopfield neural network. *Pattern Recognition, 25:2*, 197–209.

Received December 16, 2002
Revised December 16, 2002
Accepted January 12, 2004
Final manuscript February 3, 2004