Distance-based Classification of Structures within a Connectionist Framework

Brijnesh J. Jain and Fritz Wysotzki*

*Methods of Artificial Intelligence, Computer Science Department, Sekr. Fr 5–8, Technical University Berlin, Franklinstr. 28/29, D-10587 Berlin, Germany, e-mail: {bjj,wysotzki}@cs.tu-berlin.de

Abstract. We present a novel self-organizing classification network for structured objects. The system consists of two layers, an upper layer which serves as a classifier and a lower layer as a feature extractor. The classifier is an inhibitory winner-take-all network where each unit corresponds to a unique class while the lower layer consists of Hopfield-style networks solving the problem of finding maximum common parts of the input structure and the particular prototypes. The self-organizing architecture is able to outperform traditional classification models using a maximum selector as a classifier. The improvements arise from deactivating networks in the lower layer which indicate large dissimilarities between the input structure and the corresponding prototypes.

Keywords. Self-organization, distance-based classification, winner-take-all networks, graphs

1 Introduction

In many domains an appropriate and popular representation of structured objects are labeled graphs ([15]). Distance-based classification of graphs requires graph matching procedures to compute the distance or similarity of two graphs.

In general graph matching problems are well-known NP-complete problems. Thus the algorithms are useless for all but the smallest graphs, since the execution times of graph matching algorithms grow exponentially with the size of the graphs. But in practice it is often sufficient to find approximate solutions which are near the optimal solution as long as some temporal conditions are satisfied. Therefore much effort has been directed toward devising efficient heuristics to find optimal or approximate solutions for graph matching problems. Among other heuristics artificial neural networks have been proposed as a promising model of computation for solving a wide variety of combinatorial problems including graph matching problems ([6], [15], [16]).

A classification model for structured objects using neural networks consists of three parts: a transducer, a feature extractor, and a classifier. The transducer senses the input graphs and constructs suitable neural networks for matching the input with the prototypes. The feature extractor applies the neural networks to compute the proximities between the input and the particular prototype graph. The classifier uses the proximity data to assign the input graph to one of the finite number of classes represented by the corresponding prototypes. The most common and classical approach uses a maximum selector as a classifier ([14], [15]). The maximum selector classifier determines the maximum value from a set of numerical input data by directly comparing the values and finally selects the class corresponding to the largest value. The task of retrieving the most similar prototype is a basic problem not only in classification, but also in pattern recognition or case-based reasoning.

In this paper we replace the maximum selector classifier by a neural network in order to investigate a purely *self-organizing classification* (SOC) architecture and the self-organization of coupled subnets. Thus this approach is an emancipation from a homunculus who controls the feature extraction processes and finally selects the particular class.

One way to deal with the maximum selection from a set of inputs within a connectionist framework are *winner-take-all* (WTA) networks ([3]). The operation of these networks is a mode of contrast adjustment and pattern normalization where only the unit with the highest activation fires and all other units in the network are inhibited after some setting time. An example of a common competitive architecture to select the maximum or minimum from a set of data is MAXNET ([10]). Other techniques to pick a maximum can be found in ([3], [5], [9], [11]). In [12] the maximum selection is generalized to k-winners-take-all networks which identify the largest k of n real numbers. Robust WTA architectures with evidential response are considered in ([7], [8]).

The overall architecture of the self-organizing classification network consists of an upper and lower layer representing the classifier and the feature extractor, respectively. The WTA network in the upper layer serves as a decision net where each unit represents one class. Units which respond to a higher level concept are sometimes called grandmother cells[1] ([1]). In order to respond uniquely to the class of the input graph the grandmother cells are connected with the subnets of the lower level feature extractor.

As the SOC network evolves the subnets in the lower layer provide a current estimation of the similarity between the input graph and the corresponding prototype at any point of time. The feature extractor transforms and scales the estimations to processible values and feeds them as an external input into the decision net. During evolution the SOC network deactivates subnets which presumably ascertain high dissimilarity between the input graph and the corresponding prototype in order to improve performance by focusing on a few potential classes. In first experiments we compared the behavior and performance of self-organizing systems with classification models using the maximum selector.

The paper is organized as follows: Section 2 is a brief introduction in basic notions of graph theory. Section 3 describes the SOC network architecture. First experiments are discussed in Section 4. Finally, the last section summarizes this contribution and provides an outlook for further research.

2 Basic graph theory

A *simple graph* without loops and multiple edges is a pair $G = (V, E)$ consisting of a finite set $V \neq \emptyset$ of *nodes* and a binary relation $E \subseteq V^{[2]}$ where $V^{[2]} := \{\{i, j\} \mid i, j \in V,\ i \neq j\}$ is the set of pairs of elements of V. The elements $\{i, j\} \in E$ are called *edges*.

A *labeled graph* $G = (V, E_0^L)$ consists of a finite set $V \neq \emptyset$ of nodes and a set of binary relations $E^l \in E_0^L$ $(0 \leq l \leq L)$ called the l-th *label class* of G, such that

1. $m \neq n \Rightarrow E^m \cap E^n = \emptyset$
2. $V^2 = \bigcup_{i=0}^{L} E^l$
3. $(i, j) \in E^l \Leftrightarrow (j, i) \in E^l$
4. $E^l \cap \Delta_V \neq \emptyset \Rightarrow E^l \subseteq \Delta_V$

where $\Delta_V := \{(j, j) \mid j \in V\}$ is the diagonal of V^2. In a labeled graph G all nodes and all edges are assigned labels in such a way that the labels of the nodes are different from the labels of the edges. Relations $E^m \subseteq \Delta_V$ represent node labels whereas all other relations E^n are disjoint to Δ_V and represent edge labels.

An example for labeled graphs are chemical structures. Nodes of a graph represent atoms and it edges represent bonds between atoms. Node and edge labels specify the corresponding type of atoms and bonds, respectively. Missing bonds between atoms are characterized by distinguished edge labels denoting their nonexistence.

Any simple graph $G = (V, E)$ can be considered as a labeled graph $G' = (V, E_0^2)$ with three labels. Assign all nodes to label class E^0, all edges to label class E^1 and all missing edges to label class E^2. A labeled graph $G = (V, E_0^L)$ is completely described by its adjacency matrix $A_G = (e_{ij})$ with $e_{ij} = l$ if $(i, j) \in E^l$.

In the following we consider all graphs to be labeled. If we are not interested in the specific labels briefly denote with E the set E_0^L of label classes. Furthermore $E^0 := \Delta_V$ denotes the set of all node label classes, $E^1 \subseteq V^{[2]}$ the set of all edge label classes, and $E^2 \subseteq V^{[2]}$ the label class of missing edges.

Let $G = (V, E)$ be a labeled graph. For each node $i \in V$ the number of edges $|\{i, j\} \in E^1 \mid j \in V\}|$ is the *valency* of node i. If the valency of all nodes is identical we call G *regular*. A *subgraph* $H = (V_H, E_H)$ of $G = (V_G, E_G)$ is a graph with $V_H \subseteq V_G$ and $E_H^1 \subseteq V_H^{[2]} \cap E_G^1$. A graph $G = (V, E)$ is called *complete*, if $E^2 = \emptyset$. A complete subgraph $C \subseteq G$ is called *clique* of G. A *maximum clique* $C \subseteq G$ is a clique with maximum number of nodes. An *induced subgraph* $H = (V_H, E_H)$ of $G = (V_G, E_G)$ is a subgraph with $E_H^1 = V_H^{[2]} \cap E_G^1$.

Let $A_1 = (e_{ij}^1)$ and $A_2 = (e_{ij}^2)$ be the adjacency matrices of labeled graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, respectively. An *isomorphism* of G_1 and G_2 is a bijective mapping $\phi : V_1 \to V_2$ with $e_{ij}^1 = e_{kl}^2$ where $k = \phi(i)$ and $l = \phi(j)$. A *common isomorphic subgraph* $G' = (V', E')$ of G_1 and G_2 is a graph which is isomorphic to some subgraph in G_1 and in G_2. A maximal common isomorphic induced subgraph of G_1 and in G_2 is called *joint subgraph* of G_1 and G_2.

[1]The term grandmother cell is based on the folklore that everyone has a neuron that fires if and only if one sees his grandmother. This view has become largely discredited in biological circles. Yet, it is hard to deny that such neurons exists ([1]).

3 The SOC model

The SOC network consists of two interconnected layers. A subordinated (lower) layer which serves as a feature extractor and a superordinated (upper) layer as a classifier. The lower layer extracts possibly relevant information from the input graph. The upper layer processes this information to assign the input graph to one of a finite number of classes. In the following we first introduce both layers separately. Thereafter we assemble them to a complete network.

3.1 Lower layer: graph matching using the Hopfield network model

The computation of graph distances and similarities requires the computation of partial mappings between graphs. The task to find the best partial mapping between two graphs where the quality of the mapping is estimated in terms of a problem dependent objective function is called graph matching.

We regard graph matching as a superordinate concept of four classes of partial graph mappings: (GM_1) graph isomorphism, (GM_2) subgraph isomorphism, (GM_3) maximum common subgraph isomorphism, and (GM_4) maximum common induced subgraph isomorphism. (GM_1) and (GM_2) are applied as a query engine to compare a given pattern against pre-stored structure database, such as chemical structure and protein data bank, inference rule knowledge base, etc. (GM_3) and (GM_4) is frequently used to identify common structures of the objects.

To illustrate the operation of a SOC model it is sufficient to restrict ourselves to one graph matching problem. For convenience we focus on the problem (GM_4) of finding the maximum common isomorphic induced subgraph (briefly: joint subgraph) of two labeled graphs G_1 and G_2.

The Hopfield network is known to be useful as a model of computation for solving optimization problems where the network is expected to find a configuration which minimizes an energy function. Thus to solve graph matching problems using neural networks, one must cast these problems into an optimization problem. The general approach as proposed by Hopfield and Tank ([4]) maps the objective function of the optimization problem that needs to be minimized to a suitable network energy function, while the constraints of the problem are included as penalty terms.

Following this method we first construct the association graph $A(G_1, G_2)$ of G_1 and G_2. This maps the graph matching problem to the optimization problem of finding a maximum clique in $A(G_1, G_2)$, since the maximum cliques in $A(G_1, G_2)$ are in 1-1 correspondence to the joint subgraphs of G_1 and G_2.

Definition 3.1 *Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be labeled graphs with adjacency matrices $A_1 = (e^1_{ij})$ and $A_2 = (e^2_{ij})$, respectively. Then the simple graph $A(G_1, G_2) = (V, E)$ with*

$$V = \left\{(i, j) \in V_1 \times V_2 \mid e^1_{ii} = e^2_{jj}\right\}$$
$$E = \left\{\{(i, j), (k, l)\} \in V^{[2]} \mid i \neq k, j \neq l, e^1_{ik} = e^2_{jl}\right\}$$

is called the association graph *of G_1 and G_2.*

The nodes of the association graph are pairs of nodes $i \in V_1$ and $j \in V_2$ which belong to the same label class. Similarly, two nodes in $A(G_1, G_2)$ are connected with an edge if and only if the corresponding edges in the graphs G_1 and G_2 have identical labels. Note, that the association graph is a well-known auxiliary structure for solving graph matching problems (see e.g. [2]).

Next we transform the association graph $A(G_1, G_2)$ to a Hopfield network model. The dynamics of the network describes a mechanism that seeks a minimum of an energy function $\mathcal{E}$, such that the solutions of the maximum clique problem correspond to the global minima of $\mathcal{E}$.

The network model consists of a fully interconnected system of n units where n is the number $|V|$ of nodes in $A(G_1, G_2)$. The strength of the connection between unit i and unit j is determined by the synaptic weight $w_{ij} = w_{ji}$. The internal state $x_i(t)$ of each unit i at time t is equivalent to the weighted sum of the external states of all connecting units. The external state $f(x_i(t))$ of each unit i is determined by a nonlinear transfer function f which is bounded below by 0 and above by 1. The network algorithm is completely defined by the following system of equations:

$$x_i(t+1) = (1-d)x_i(t) + \sum_{j \neq i} w_{ij} f(x_j(t)) \quad (1)$$

where $d \in [0, 1]$ describes the self-inhibition of the units. The algorithm thus proceeds as follows. An input vector is imposed on the network as its initial state. Finding a maximum clique then proceeds by updating the internal state $\boldsymbol{x}(t+1)$ in accordance with the update rule (1). The updating procedure is repeated until the network reaches a stable state which corresponds to a clique of $A(G_1, G_2)$. The clique size can be read out by counting the number of active units, i.e. the number of units with activation $x_i(t) \geq 1$.

To compute solutions for the maximum clique problem appropriate weights must be selected to map the corresponding objective function to the energy function of the network. In general the weights are adjusted according to the following rule:

$$w_{ij} = \begin{cases} w_E > 0 & : \text{ if } (i,j) \in E \\ -w_I < 0 & : \text{ otherwise} \end{cases}$$

for all units $i, j \in V$. Connections with positive weight w_E (negative weight w_I) are called excitatory (inhibitory) connections. Inhibitory connections restrain inconsistent hypotheses and therefore are responsible to ensure feasibility of the solution. On the other hand, excitatory connected units mutually reinforce the possible hypotheses.

Many variations of the Hopfield network have been proposed for ensuring feasibility and improving the solution quality through escape from local minima of the energy function ([16]). For our purposes we choose as a representative model a variant of the algorithm from ([18]), since it is simple and it has not only been applied to classical optimization problems, but also to practical classification and knowledge discovery tasks in chemistry ([14], [15]). Thus, we make the following design decisions: The transfer function f is the piecewise linear limiter function

$$f(x) = \begin{cases} 1 & : \text{ if } x \geq 1 \\ x & : \text{ if } x \in \,]0,1[\\ 0 & : \text{ if } x \leq 0 \end{cases} \tag{2}$$

with an upper and lower saturation point. The self-inhibition d is set to zero, such that each unit i has a self-excitation of $w_{ii} = 1$. A convergence analysis of this network model is given in [13].

3.2 Upper layer: the decision net

In distance-based classification problems a classifier assigns an input pattern to the class for which the corresponding discriminant is largest. Here the discriminant functions are a given proximity measure. Thus the class decision boils down to select the maximum value of a set of data.

Several different neural networks can perform the selection of a maximum value from a set of inputs. A common architecture are inhibitory WTA networks which we will employ as a self-organizing classifier. To this end let N be the number of distinct classes $C_1, \ldots, C_N$. The network consists of N mutually inhibitory connected units, such that each unit uniquely represents a class. The network evolves according to the update rule

$$y_i(t+1) = (1-d)y_i(t) - w \sum_{j \neq i} y_j(t) + I_i(t) \tag{3}$$

where $y_i(t)$ is the activation of unit i, $-w < 0$ represents the joint inhibitory strength of the synapses connecting any pair of units, $d > 0$ is the self-inhibition and $I_i(t)$ an external input. The function f is either the piecewise linear limiter transfer function as given in (2) or a semi-linear transfer function

$$f(y) = \begin{cases} 0 & : \text{ for } y < \theta \\ \kappa(y-\theta) & : \text{ for } y \geq \theta \end{cases} \tag{4}$$

where $\kappa > 0$ is the gain of f. In contrast to the limiter function (2) the upper saturation limit is missing in the semi-linear transfer function (4). In case of the limiter function (2) its lower and upper saturation limits bound the dynamics of the net. Similarly, under specific conditions the non-divergence of networks with non-saturating semi-linear transfer functions is established ([17]).

The aim of this network is to discriminate the components of an input vector. The input vector may be fed in either by an external input layer into the decision net or by initializing the units of the decision net accordingly.

In general a WTA net for maximum selection is designed to signal the particular unit to select where information about the evidence in the decision made is missing. This information would be useful to arrange for ambiguous decisions and thereby improve the classification performance of the network. Furthermore WTA networks are not robust to failure of single units or noisy data. If a unit or its connecting links are damaged, performance is impaired or faulty and the network behaves unreliable. In classification tasks or prototype detection, failure of one WTA unit means loss of the whole class. Provided that the external input is constant both drawbacks can be removed by incorporating distributed redundancy and differentiating units into the network. These extensions lead to a system which is not only robust against failure of single units but also efficiently sharpens the focus on the given problem in terms of a faster generated and more accurate evidential response ([7], [8]).

3.3 The SOC network model

We have so far been concerned with the particular building blocks of the SOC network model, namely the feature extractor in the lower layer and the classifier in the upper layer. In this section we assemble both components by linking them together to a system of coupled subnets.

Let G be an input graph and $P_1, \ldots, P_N$ be N prototype graphs of class $C_1, \ldots, C_N$, respectively. Figure 1 shows the architecture of a SOC network for a 3-class problem ($N = 3$). The lower layer has N subnets $S_1, \ldots, S_N$. Each subnet S_k computes the common joint subgraph of input G and the k-th prototype P_k. The upper layer contains a WTA network with N

mutually inhibitory connected units where unit k represents class C_k. Each subnet S_k of the lower layer is coupled with unit k of the top level network. The links interconnecting the devices of both layers represent the external inputs $I_k(t)$ in (3) fed into the decision net.

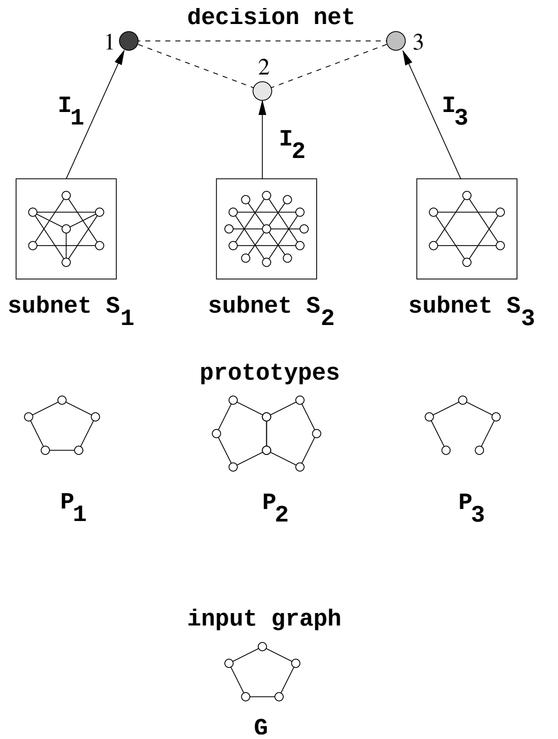

Figure 1   Architecture of a SOC network.

The SOC network is completely described by the following system of equations

$$\begin{aligned} y_k(t+1) &= (1-d)y_k(t) - w\sum_{j\neq k} y_j(t) + I_k(t) \\ I_k(t+1) &= F(\boldsymbol{W}_k, \boldsymbol{x}^k(t)) \\ x_{\boldsymbol{i}}^k(t+1) &= (1-d)x_{\boldsymbol{i}}^k(t) + \sum_{j\neq i} w_{\boldsymbol{ij}} f(x_{\boldsymbol{j}}^k(t)) \end{aligned}$$

where $y_k(t)$ is the activation of the k-th unit in the decision net and $x_{\boldsymbol{i}}^k(t)$ is the activation of the $\boldsymbol{i}$-th unit in the k-th subnet S_k. The first and the last equation correspond to the update rules (3) and (1) of the upper and lower layer, respectively. The external input $I_k(t)$ is a function F of the current state $\boldsymbol{x}^k(t)$ of the subnet S_k and the subnet itself which is represented by its weight matrix $\boldsymbol{W}_k$. The evolution process of the network terminates when only one unit fires and all other units are inhibited.

One iteration step of the SOC network model simultaneously updates the subnets in the lower layer. A current similarity value of the updated state of the subnet is extracted and sent as an external input to the corresponding unit in the upper layer. The current similarity value reflects the similarity of the underlying graphs at each iteration step with respect to a given distance or similarity measure. Finally, using the external input the system updates the units in the upper layer.

The SOC network inhibits response tendencies of subnets which seem to drift to states corresponding to dissimilarities of the graphs to compare. If the activation of a complex unit k declines to zero the subnet S_k is deactivated prematurely. The idea is to focus on promising subnets which tend to reveal high similarity between the graphs to compare and to neglect superfluous computations of matching dissimilar graphs.

Crucial for the performance of a SOC network is the choice of the weight w of inhibitory connections in the upper layer and the external input. The inhibition w of the decision net controls the selection pressure. High inhibition results in early decisions while low inhibition corresponds to low selection pressure and therefore to a longer decision making. The external input comprises the current similarity value and an attention parameter. The attention parameter controls the significance of the current similarity value in the decision making.

In the following we introduce a possible choice of an external input based on the Zelinka distance

$$d(G_1, G_2) = \max\{|V_1|, |V_2|\} - |V_U|$$

where $G_i = (V_i, E_i)$ are two graphs $(i = 1, 2)$ and $U = (V_U, E_U)$ is a joint subgraph of G_1 and G_2. The Zelinka distances between the input graph G and the prototypes P_1, P_2, and P_3 given in Figure 1 are $d(G, P_1) = 0$, $d(G, P_2) = 3$, and $d(G, P_3) = 1$, respectively. Thus we expect that the SOC network assigns G to class C_1 as indicated by the darkest unit in the upper layer.

In order to obtain a current similarity value we map the Zelinka distance d to the total connective intensity of the subnets in the lower layer. Let $A_k := A(G, P_k) = (V_k, E_k)$ be the association graph of input graph G and the k-th prototype P_k. Then the total connective intensity $\omega_k(t)$ of subnet S_k is defined as

$$\omega_k(t) = \frac{1}{2}\left(\sum_{(\boldsymbol{i},\boldsymbol{j})\in E_k} x_{\boldsymbol{i}}^k(t) + x_{\boldsymbol{j}}^k(t) \;-\; \sum_{(\boldsymbol{i},\boldsymbol{j})\notin E_k} p_{\boldsymbol{ij}}^k(t)\right)$$

where the scaling factor $1/2$ is included to count each edge once. The first term sums the activations of adjacent units for each excitatory connection. The second term is a penalty term which imposes a penalty $p_{\boldsymbol{ij}}^k(t) \geq 0$ for each inhibitory edge connecting units with positive activations $x_{\boldsymbol{i}}^k(t), x_{\boldsymbol{j}}^k(t) > 0$. The penalty is a monotonously increasing function of t (see Section 4). The intention to introduce a penalty term is to consider only feasible solutions of the graph matching procedure. Thus in a stable state the penalty term is zero and the total connective intensity is the number of edges of the clique found by subnet S_k.

Note, that by construction of the subnet architectures in the lower layer the number of excitatory edges between active units is maximal if and only if these units constitute a maximum clique. Though it would be more natural to consider active units only, the class decision is susceptible to deceptions by means of large total activations of particular subnets containing alternative hypotheses. Therefore the current similarity values take the specific type of connections between active units into account.

Using the total connective intensity, the external input $I_k(t)$ is expressed by

$$I_k(t) = \gamma_k(t) f\left(\frac{\omega_k(t)}{\mu_k}\right) \tag{5}$$

where $0 \leq \gamma_k(t) \leq 1$ is an attention parameter, the function f is the limiter function as defined in (2), and $\mu_k := (|V_k|^2 - |V_k|)/2$ is the number of edges of a complete graph with $|V_k|$ nodes.

The attention parameter $\gamma_k(t)$ controls the significance of the current state of subnet S_k with regard to the classification task. Therefore $\gamma_k(t)$ should be monotonously increasing with increasing time t. The connective intensity $\omega_k(t)$ is normalized by μ_k to take the different sizes of the prototype graphs into account. If the penalty term in $\omega_k(t)$ is too small, during evolution of S_k the intensity $\omega_k(t)$ can exceed the maximum feasible value μ_k. To this end the normalized connective intensity is bounded by f.

4 Experiments

For testing the behavior of a self-organizing classification network we used two test series of graphs as depicted in Figure 2 and 3 as representative prototypes of simple multi-class problems.

The first test series contains all possible cubic graphs with eight nodes, i.e. all regular graphs with valency three and order eight. The second series consists of four simple graphs of different size with maximum valency three.

Parameter settings: Throughout this paragraph we exceptionally drop the class index k running from 1 to N. To this end let $G = (V_G, E_G)$ be an input graph and $P = (V_P, E_P)$ be a prototype. With S we denote the subnet matching G and P.

The general parameter setting for a subnet S in the

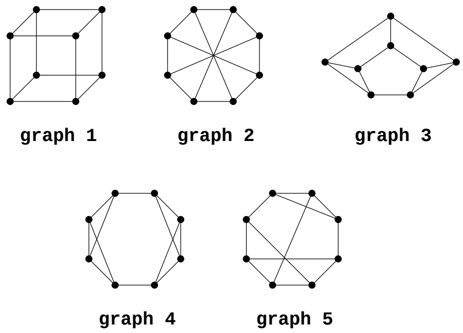

Figure 2 Test series 1: Cubic graphs of order 8

lower feature extraction layer follows a theoretical analysis given in [13]. For the excitatory weight we set

$$w_E = \frac{2}{\deg_E \cdot \deg_I + \deg_E}$$

where $\deg_E$ ($\deg_I$) is the maximum valency of an unit with respect to its excitatory (inhibitory) edges. For the inhibitory weights we define a deterministic annealing schedule to improve the solution quality and to guarantee feasibility. Starting with a small value $c = w_E/deg_I^2$ the inhibitory weight is adapted in each iteration step according to the rule $w_I(t+1) = w_I(t) + c$. In order to arrive at a stable state where the output vector $f(\boldsymbol{x}(t))$ lies in a corner of the unit hypercube, instable equilibriums points in the interior of the hypercube, which correspond to ambiguities, are resolved by imposing noise on the corresponding units.

The following parameter settings are chosen heuristically within considerations of plausibility. The external input is given by (5) with attention parameter

$$\gamma(t) = f\left(\frac{\nu_{\min}}{2\,|S|^2 \cdot \nu_{\max}} \cdot t\right) \tag{6}$$

where $\nu_{min} = \min\{|V_G|, |V_P|\}$ and $\nu_{\max} = \max\{|V_G|, |V_P|\}$ are the minimum and maximum number of nodes of G and P. The value $|S|$ is the number of units of subnet S and f is the limiter function as defined in (2). The limiter function f bounds the activation to a limited capacity. The first term inside f brings the attention into sharper focus for smaller subnets with similar size of the graphs. In order to synchronize the attention among all subnets in the lower layer we incorporate the size $|S|$ of the subnet into γ.

The penalty is defined by

$$p_{ij}(t) = \begin{cases} t & : \text{ if } x_i \cdot x_j > 0 \text{ and } w_{ij} = w_I \\ 0 & : \text{ otherwise} \end{cases}$$

Finally, the selective pressure expressed by the weight w of the inhibitory connections in the upper classifier layer is set to $w := 1/(N-1)$.

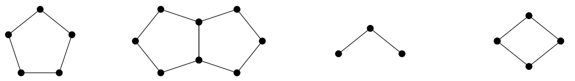

Figure 3 Test series 2: Graphs of max. valency 3

Implementation: The graphs in both test series serve as prototype graphs. Thus in the first case we have a 5-class problem and in the second case a 4-class problem. For the first test series we perform five and for the second test series four runs of the SOC network, such that in the i-th run the i-th prototype graph is also the input graph.

To evaluate the performance of a SOC network we also consider the same architecture using the maximum selector classifier in the upper layer instead of a self-organizing competitive decision network. In contrast to the purely self-organizing system all subnets evolve until they arrive in a stable state. Then the maximum selector classifies the input graph.

Results: Again k is assigned to the class index running from 1 to N. Let $t_{1,k}$ denote the number of iterations of subnet S_k until deactivation or termination. Unit k in the decision network deactivates S_k if the activation $y_k(t)$ of the k-th unit equals zero. Furthermore let $t_{2,k}$ be the number of iterations until S_k attains stability. Note, that always $t_{1,k} \leq t_{2,k}$ and equality holds if S_k is stable while unit k in the decision net is still active. For $i \in \{1,2\}$

$$\bar{t}_i = \frac{1}{N}\sum_{k=1}^{N} t_{i,k}$$

denotes the average number of iterations averaged over all N subnets S_k. Since $t_{1,k} \leq t_{2,k}$ we are interested in the total amount of iterations Δ we save using a self-organizing classifier instead of the maximum selector. The difference Δ is given by

$$\Delta = \sum_{k=1}^{N} t_{2,k} - t_{1,k}$$

With $r = \bar{t}_1/\bar{t}_2$ we denote the percentage of the total amount of iterations using a SOC network in relation to the classical approach using a maximum selector.

Table 1 and 2 summarize the results of a selected test cycle for both test series. Each row in the tables shows the input graph P_k, the activation of the winning unit $y_k(t_1)$ in the decision net, the average number of iterations of all subnets $\bar{t}_1$, the average number of iterations $\bar{t}_2$, the difference Δ, and the percentage r of expenses using a SOC network.

In all runs the subnets found a maximum clique of isomorphic graphs and in most cases they were capable of finding a joint subgraph. Only in a few cases approximate solutions were found. Nevertheless in all runs the SOC network correctly classified the input graphs.

Test series 1: Although all prototype graphs have the same number of nodes, the same number of edges, and the same valency sequence the behavior of the SOC network is noticeably varying. In the best case the SOC network saves 66% and in the worst case 20% iterations in comparison to the maximum selector approach.

	$y_k(t_1)$	$\bar{t}_1$	$\bar{t}_2$	Δ	r
P_1	0.53	3181.0	7030.2	19256	0.45
P_2	0.62	2109.8	6146.6	20178	0.34
P_3	0.64	2914.0	6034.2	15596	0.48
P_4	0.83	7547.4	9506.6	9793	0.79
P_5	0.69	6752.8	8444.2	8451	0.80

Table 1 Results of test series 1

Test series 2: Most noticeably is the negligible improvement of the SOC network compared to the maximum selector classifier if the input is P_2. This observation is opposed to the efficient performance of the SOC network in all other cases. In the average the SOC network requires only 29% of the computational effort required by the maximum selector classifier.

	$y_k(t_1)$	$\bar{t}_1$	$\bar{t}_2$	Δ	r
P_1	0.40	420.0	1193.5	3094	0.35
P_2	0.40	2672.5	2909.0	946	0.92
P_3	0.48	130.0	611.5	1926	0.21
P_4	0.43	231.0	722.3	1965	0.32

Table 2 Results of test series 2

Discussion: Of course, an experimental investigation on just two test series does not allow general conclusions. Rather we intend to provide an impression of the SOC network on a small set of test cases.

The results in Table 1 and 2 show that a SOC network is capable to outperform the traditional classification approach using the maximum selector classifier. But further test runs reveal that the SOC network behaves unreliable in the sense, that it is not robust against misclassifications like the maximum selector classifier. Even though the subnets find a joint subgraph the SOC network may assign the input graph to a *wrong* class. The reason is that during evolution to a final state the intensity as well as the connective intensity of individual subnets noticeably exceed the intensities of the other subnets for a short-time. This may lead to deceptions and finally to wrong class decisions if the attention parameter and the selection pressure are not chosen appropriately. Another problem is to adjust the attention parameters in order to synchronize the states of the subnets in the feature extractor layer.

5 Conclusion

In this contribution we presented a self-organizing classification network for structured objects. The network consists of a feature extractor layer and a classifier layer which are interconnected in a feed-forward manner. The feature extractor contains Hopfield-style networks to compare the input graph with the prototypes and the classifier is an inhibitory WTA network. This architecture leads to freedom from a homunculus who controls and observes the evolving subnets until they all arrive in a stable state and then finally selects the particular class.

In contrast to classification models using the maximum selector classifier a self-organizing classification model takes transient states during the matching process into account. This approach favors matchings which promise higher similarity between the underlying structures and deactivates subnets which tend to indicate high dissimilarities. Altogether this strategy outperforms the maximum selector classifier and significantly reduces the effort. The improved performance of the SOC network is achieved by suffering losses in the reliability with respect to correct class decisions.

Further research comprises a theoretical analysis of the dynamics of the SOC network. Within a theoretical foundation appropriate parameter settings and estimations for the attention and the selection pressure may be derived to reduce the rate of misclassifications and to further improve performance.

References

1. D.J. Amit. *Modeling Brain Function: The world of attractor neural networks*. Cambridge University Press, 1989.
2. D.H. Ballard and C.M. Brown. *Computer Vision*. Prentice Hall, Englewood Cliffs, New Jersey, 1982.
3. J.A. Feldmann and D.H. Ballard. Connectionist models and their properties. *Cognitive Science*, 6:205–254, 1982.
4. J.J. Hopfield and D.W. Tank. Neural computation of decisions in optimization problems. *Biological Cybernetics*, 52:141–152, 1985.
5. J.J. Hopfield and D.W. Tank. Computing with neural circuits: A model. *Science*, 223:625–633, 1986.
6. A. Jagota. Approximating maximum clique with a hopfield network. *IEEE Trans. Neural Networks*, 6:724–735, 1995.
7. B.J. Jain and F. Wysotzki. Efficient pattern discrimination with inhibitory WTA nets. In *Proc. ICANN'01*, 2001. To appear.
8. B.J. Jain and F. Wysotzki. On the short-term-memory of WTA nets. In M. Verleysen, editor, *Proc. ESANN'01*, pages 289–294. D-Facto, Brussels, 2001.
9. T. Kohonen. *Self-Organization and Associative Memory*. Springer-Verlag, Berlin, 1984.
10. R.P. Lippman. An introduction to computing with neural nets. *IEEE ASSP Magazine*, pages 4–22, April 1987.
11. R.P. Lippman, B. Gold, and M.L. Malpass. A comparison of hamming and hopfield neural nets for pattern classification. Technical Report TR-769, MIT Lincoln Laboratory Technical Report, 1988.
12. E. Majani, R. Erlanson, and Y. Abu-Mostafa. On the k-winner-take-all network. *Advances in Neural Information Processing Systems*, 1:634–642, 1989.
13. K. Schädler and F. Wysotzki. Theoretical foundations of a special neural net approach for graphmatching. Technical Report 96-26, Dept. of Computer Science, TU Berlin, 1996.
14. K. Schädler and F. Wysotzki. Application of a neural net in classification and knowledge discovery. In M. Verleysen, editor, *Proc. ESANN'98*, pages 117–122. D-Facto, Brussels, 1998.
15. K. Schädler and F. Wysotzki. Comparing structures using a Hopfield-style neural network. *Applied Intelligence*, 11:15–30, 1999.
16. K.A. Smith. Neural networks for combinatorial optimisation: A review of more than a decade of research. *INFORMS Journal on Computing*, 11(1):15–34, 1999.
17. H. Wersing, J.J. Steil, and H. Ritter. A competitive layer model for feature binding and sensory segmentation. *Neural Computation*, 13(2):357–387, 2001.
18. F. Wysotzki. Artifical intelligence and artifical neural nets. In *Proc. 1st Workshop on AI*, Shanghai, September 1990. TU Berlin and Jiao Tong University Shanghai.