On the Efficient Classification of Data Structures by Neural Networks

Paolo Frasconi
Dipartimento di Sistemi e Informatica
Universita di Firenze
Via di Santa Marta 3
50139 Firenze, Italy

Marco Gori
Dipartimento di Ingegneria deU'Informazione
Universita di Siena
Via Roma 56
53100 Siena, Italy

Alessandro Sperduti
Dipartimento di Informatica
Universita di Pisa
Corso Italia, 40
56125 Pisa, Italy

Abstract

In the last few years it has been shown that recurrent neural networks are adequate for processing general data structures like trees and graphs, which opens the doors to a number of new interesting applications previously unexplored. In this paper, we analyze the efficiency of learning the membership of DO AGs (Directed Ordered Acyclic Graphs) in terms of local minima of the error surface by relying on the principle that their absence is a guarantee of efficient learning. We give sufficient conditions under which the error surface is local minima free. Specifically, we define a topological index associated with a collection of DOAGs that makes it possible to design the architecture so as to avoid local minima.

1 Introduction

It is well-known that connectionist models are not only capable of dealing with static patterns, but also with sequential inputs. Real world, however, often proposes structured domains that can hardly be represented by simple sequences. For instance, there are cases in which the information can be framed naturally in graphs of variable size, and one may be interested in processing these structures as a whole, and not pay attention specifically to their nodes. The ability to classify these structured data is fundamental in a number of different applications such as medical and technical diagnoses, molecular biology and chemistry, automated reasoning, software engineering, geometrical and spatial reasoning, and pattern recognition. Neural networks for processing data structures have been proposed by Pollack [1990] and, recently, by Sperduti, Starita & Goller [1995], and by Sperduti & Starita [1997]. It has been shown that they can actually be used for classifying data structures by using an algorithm, referred to as BPTS (backpropagation through structure), that extends naturally the time unfolding carried out by BPTT, in the case of sequences. It has also been pointed out that BPTS is significantly better suited for dealing with long-term dependencies than BPTT, because of its inherent unfolding through structures instead of simple lists, the data structure counterpart of sequences. As for any neural network learning algorithm, however, the efficiency of *BPTS,* may be seriously plagued by the presence of local minima in the associated error function. In the epilogue of the expanded edition of Perceptron, Minsky [1988] pointed out that

"... as the field of connectionism becomes more mature, the quest for a general solution to all learning problems will evolve into an understanding of which types of learning processes are likely to work on which classes of learning problems. And this means that, past a certain point, we won't be able to get by with vacuous generalities about hill-climbing. We will really need to know a great deal more about the nature of those surfaces for each specific realm of problems that we want to solve."

In this paper, we analyze the efficiency of learning the membership of DOAGs (Directed Ordered Acyclic Graphs) in terms of local minima of the error surface by relying on the principle that their absence is a guarantee of efficient learning. We give a sufficient condition under which the error surface is local minima free. In particular, we define a topological index associated with a collection of DOAGs that make it possible to design the architecture so as to avoid local minima. The condition we give holds for any training set composed of graphs with symbolic nodes and a neural network capable of learning the assigned data.

2 Recurrent networks for processing of data structures

In this section, we review briefly the basic idea proposed in [Sperduti and Starita, 1997] concerning adaptive processing of data structures. In particular, we focus on the classification of DOAGs (Directed Ordered Acyclic Graphs) [Arbib and Given'on, 1968] and show that their membership can be learned from examples. The recurrent networks considered in this paper process data structures beginning from a fixed initial state. Each experiment $\mathcal{E}$ involving data structures can conveniently be expressed by means of three entities: a learning environment $\mathcal{L}_e$ (set of data used for learning), a network $\mathcal{N}$, and a cost index E.

- *Learning Environment* $\mathcal{L}_e$.

  Instances in the learning domain are structured pieces of information described by annotated directed ordered acyclic graphs (DOAGs). Here for a DOAG we mean a DAG $S \doteq \{V, E\}$ with vertex set V and edge set E, where for each vertex $v \in V$ a total order on the edges leaving from v is defined. Specifically, let $\mathcal{O}_p \doteq \{q : (v_p, v_q) \in E\}$ and $O_p \doteq |\mathcal{O}_p|$ the outdegree of node v_p, then the *out-port function* $\rho(\mathcal{O}_p, q)$ returns the position of the edge (v_p, v_q) in the total order defined on the edges leaving from v_p.[1] Similarly, let $\mathcal{I}_p \doteq \{q : (v_q, v_p) \in E\}$, and $I_p \doteq |\mathcal{I}_p|$ the indegree of node v_p, then $\sigma(\mathcal{I}_p, q)$ is the *in-port function* which returns the position of the edge (v_q, v_p) in the total order defined on the edges leaving from v_q.

  We shall require the DOAG to possess a supersource, i.e. a vertex $v_s \in V$ such that every vertex in V can be reached by a directed path starting from v_s. The reasons for this requirement are related to the processing scheme that will be defined in the following. Note that if a DOAG does not possess a supersource, it is still possible to define a convention for adding an extra vertex v_s (with a minimal number of outgoing edges), such that v_s is a supersource for the expanded DOAG. The graphs we are considering are annotated by labels on vertices, where $\mathbf{u}_{lj}$ denotes the label attached to vertex v_j of DOAG $\mathcal{U}_l$. In the following, depending on the context, we will use $\mathbf{u}_{lj}$ to refer either a vertex or the label attached to the vertex.

  The learning environment is a collection of pairs composed of DOAGs with their own targets. Formally, let $d^-, d^+ \in R$ be such that $[d^-, d^+] \subset [\underline{d}, \overline{d}]$ and define:

  $$\mathcal{L}_e \doteq \{(\mathcal{U}_l, d_l),\ l = 1, \ldots, L\}, \qquad (1)$$

  where $\mathcal{U}_l \in \mathcal{S}_U$ is a DOAG with corresponding target value $d_l \in \{d^-, d^+\}$, and $\mathcal{U}_l \doteq \{(\mathbf{u}_{lp}, \mathbf{u}_{lq}) : \mathbf{u}_{lp}, \mathbf{u}_{lq} \in \mathcal{R}^m\}$. The collection of DOAGs $\mathcal{D}_L \doteq \{\mathcal{U}_1, \ldots, \mathcal{U}_L\}$ will be also useful itself for analysis on graph topology. A *topological sorting* $T_s^l : [1, \ldots, P_l] \to [1, \ldots, P_l] : k \to p_k$ can be associated with any $\mathcal{U}_l \in \mathcal{S}_U$ such that $\forall\ h, k : \mathbf{u}_{p_h} \prec \mathbf{u}_{q_k} \implies p_h < q_k$. Let $P_M \doteq \max P_l$ and $\mathcal{M}_{P_M}$ be the space of all matrices in $\mathcal{R}^{m, P_l}$, being $l \le L$. Using this sorting we can associate DOAGs with matrices as follows

  $$T_s : \mathcal{S}_U \to \mathcal{M}_{P_M} : \mathcal{U}_l \to \mathbf{U}_l, \qquad (2)$$

  where $\mathbf{U}_l^p \doteq \mathbf{u}_{lp}$ denotes the p-th $\mathbf{U}_l$'s column. The learning environment $\mathcal{L}_e$ can be partitioned into the following sets

  $$\begin{aligned} \mathcal{C}^+ &\doteq \{\mathcal{U}_l \in \mathcal{L}_e : \ d_l = d^+\}, \\ \mathcal{C}^- &\doteq \{\mathcal{U}_l \in \mathcal{L}_e : \ d_l = d^-\}, \end{aligned} \qquad (3)$$

  which collect the positive and the negative structures of the learning environment, respectively.

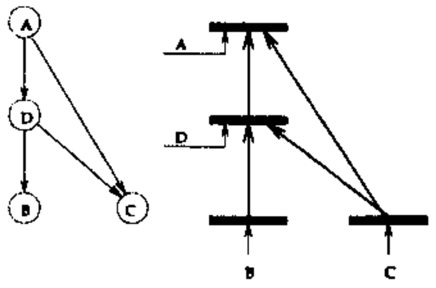

Figure 1: The *encoding network* associated with a given DOAG. The recurrent network is *unfolded through the structure* of the given DOAG.

- *Network* $\mathcal{N}$.

  The network can be thought of as a triple $\mathcal{N} \sim \{\mathbf{W}, \overline{\mathbf{V}}, \mathbf{M}\}$ of weight matrices. $\mathbf{W} \in \mathcal{R}^{n, m+1}$ is the input-neuron matrix which also incorporates biases. Note that, m is the number of input units, while n is the number of hidden units. $\overline{\mathbf{V}} \doteq \{\mathbf{V}_1, \ldots, \mathbf{V}_O\}$ are the *pointer matrices* that make it possible to extend the typical recurrent processing of sequences to data structures as put forward in the following. O is the maximum outdegree of all the nodes of the graphs belonging to $\mathcal{L}_e$ and it represents a bound on the maximum number of pointer matrices, that is $\forall r = 1, \ldots O,\ \ \mathbf{V}_r \in \mathcal{R}^{n,n}$.

  For any graph $\mathcal{U}_l$ we define the associated *encoding network* $\mathcal{N}(\mathcal{U}_l)$ as a feedforward network created as follows (see Fig. 1):

  1. The architecture of $\mathcal{N}(\mathcal{U}_l)$ is the DOAG inherited by $\mathcal{U}_l$ by simple inversion in the graph arrows.

[1] An example of out-port function is simply PASCAL **ord** function which returns the position of a given element in an ordered set. For instance $(\{3, 6, 7, 11, 13\}, 11) \to 4$.

2. Each vertex v_p $(p = 1, \ldots, P_l)$ of the graph $\mathcal{U}_l$ is associated with a corresponding layer p of n sigmoidal neurons.
3. The output of the neuron i in layer p is related to its activation a_{ilp} as follows
$$x_{ilp} = f(a_{ilp}), \qquad (4)$$
where $f(\cdot) : R \to [\underline{d}, \overline{d}]$ is a C^2 bijection such that $\forall\ a_{ilp},\ \ f'(a_{ilp}) > 0$.
4. The generic layer p of network $\mathcal{N}(\mathcal{U}_l)$ receives connections from both the input $\mathbf{u}_{lp}$ and layers with index in $\mathcal{O}_p$. Specifically, the outputs from units in layers with index in $\mathcal{O}_p$ are properly weighed by matrices $\mathbf{V}_r$ according to the *out-port function* $r = \rho(\mathcal{O}_p, q)$, $q \in \mathcal{O}_p$.
5. The computation carried out at layer p depends on matrices $\mathbf{W}$ and $\overline{\mathbf{V}}$ according to:
$$\mathbf{a}_{lp} = \sum_{q \in \mathcal{O}_p} \mathbf{V}_{\rho(\mathcal{O}_p, q)}\ \mathbf{x}_{lq} + \mathbf{W}\ \mathbf{u}^b_{lp}, \qquad (5)$$
being $\mathbf{a}_{lp} \doteq [a_{1lp}, \ldots, a_{nlp}]'$, and $\mathbf{u}^b_{lp} \doteq [\mathbf{u}'_{lp}, 1]'$ the input, taking into account also an eventual bias term.
6. When feeding the network with graph $\mathcal{U}_l$, the output for each vertex v_p is
$$o_{lp} = f(a_{lp}) = f(\mathbf{M}'\ \mathbf{x}^b_{lp}) \qquad (6)$$
where $\mathbf{x}^b_{lp} \doteq [\mathbf{x}'_{lp}, 1]'$, takes into account also an eventual bias term.

Given a structure $\mathcal{U}_l$ and the associated encoding network $\mathcal{N}(\mathcal{U}_l)$, the corresponding processing is carried out by the typical forward step of feedforward networks. Note that, apart from the architecture, which is inherited by the given DOAG, weight matrices $\mathbf{W}, \overline{\mathbf{V}}$, and $\mathbf{M}$ are independent of the given graph. Hence, the computation defined by equations (5) can be carried out by means of a *generalized recurrent network* whose computation takes place in the *pseuo-time* space $[1, \ldots, \max_l P_l]$. Let T^l_s be the topological sorting of graph $\mathcal{U}_l$ defining which node must be processed at pseudotime $p \in [1, \ldots, \max_l P_l]$. The computation described by equation (5) can be considered as a recurrent computation at pseudo-time p, once the computation had already taken place $\forall q < p$. The computation ends for $p = s = P_l$; at that pseudo-time the supersource layer activations are computed and, finally, the DOAG's membership is given by (6).

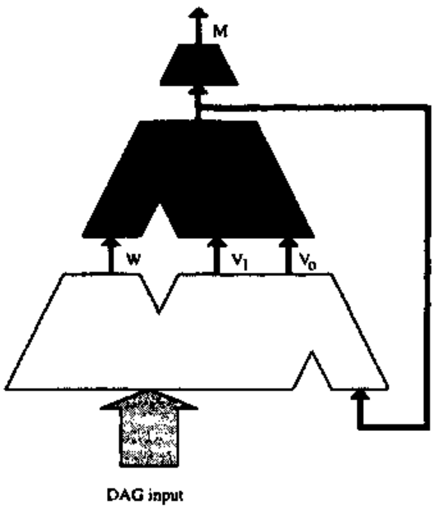

Figure 2: The generalized recurrent network architecture for processing data structures; the processing takes place in the extended time space dimension. The neuron activations are recorded in the lower module, which is also responsible for feeding the layer of neurons, according to the defined topological sorting.

- *Cost index.*

  Given the pair $(\mathcal{N}, \mathcal{L}_e)$, the output-target data fitting is measured by means of the cost function
$$E \doteq \sum_{l=1}^{L} E_l = \sum_{l \in \mathcal{C}^+} \beta_+(o_{ls} - d^+) + \sum_{l \in \mathcal{C}^-} \beta_-(o_{ls} - d^-). \qquad (7)$$
  where
$$\begin{cases} \beta_+(\alpha) = 0 & \text{if } \alpha \geq 0 \\ \beta_+(\alpha) > 0 \quad \beta'_+(\alpha) < 0 & \text{if } \alpha < 0 \end{cases}, \qquad \begin{cases} \beta_-(\alpha) = 0 & \text{if } \alpha \leq 0 \\ \beta_-(\alpha) > 0 \quad \beta'_-(\alpha) > 0 & \text{if } \alpha > 0 \end{cases}, \qquad (8)$$
  and " $'$ " stands for differentiation with respect to α. This threshold-LMS error has been introduced by Sontag & Sussman [1989]. This cost does not penalize outputs "beyond" the target values [2].

Let us introduce some more notation for gaining a compact vectorial of the gradient of the cost.

[2]It can be proven that the choice of this type of functions makes it possible to avoid the *spurious* local minima arising when choosing non-asymptotical values for the targets, that when $[d^-, d^+] \subset [\underline{d}, \overline{d}]$, that is $d^- \neq \underline{d}$ and (or) $d^+ \neq \overline{d}$. This kind of spurious local minima were shown in [Brady *et al.*, 1989], while Sontag & Sussman [1989] proved that they disappear when choosing threshold-LMS functions. Gori & Tesi [1992] proved that no such spurious local minima arise when $[d^-, d^+] \subset [\underline{d}, \overline{d}]$.

1. $\mathbf{U} \doteq [\mathbf{U}_1, \cdots, \mathbf{U}_L] \in \mathcal{R}^{m,P^*}$ collects the topological sorting of all the given data structures, where $P^* \doteq \sum_{l=1}^{L} P_l$. Similarly, for each graph $\mathcal{U}_l$, the outputs of the hidden units can be collected in matrix $\mathbf{X}_l \doteq [\mathbf{x}_{l,1}, \ldots, \mathbf{x}_{l,P_l}] \in \mathcal{R}^{n,P_l}$ and, for all the graphs, matrix $\mathbf{X} \doteq [\mathbf{X}_1, \ldots, \mathbf{X}_L] \in \mathcal{R}^{n,P^*}$ is used for keeping the hidden output trace.
2. Let us define $y_{ilp} \doteq \partial E / \partial a_{ilp}$. For node p of a given DOAG $\mathcal{U}_l$, the corresponding delta error y_{ilp} can be collected in vector $\mathbf{y}_{lp} \doteq [y_{1lp}, \ldots, y_{nlp}]'$ and the contributions from all the graph's nodes can be collected in matrix $\mathbf{Y}_l \doteq [\mathbf{y}_{l,1}, \ldots, \mathbf{y}_{l,P_l}] \in \mathcal{R}^{n,P_l}$. Finally, $\mathbf{Y} \doteq [\mathbf{Y}_1, \ldots, Y_L] \in \mathcal{R}^{n,P^*}$ contains the delta errors for all the graphs of the learning environment, while the delta error corresponding with the output unit is denoted by [3] $y_{ls} \doteq \frac{\partial E}{\partial a_{ls}}$.

The gradient of the cost can be calculated by using Backpropagation in each encoding network (see e.g. Fig. 1), that is by propagating the error through the given structure, instead of through time, as typically happens for recurrent networks processing sequences. The gradient of the cost can be written in a compact form by using the vectorial notation $\mathbf{G}_W \doteq \left[\frac{\partial E}{\partial w_{ij}}\right] \in \mathcal{R}^{m,n}$ and $\forall r \in [1, \ldots, O]$: $\mathbf{G}_{V_r} \doteq \left[\frac{\partial E}{\partial v_{ijr}}\right] \in \mathcal{R}^{n,n}$. Based on these definitions, $\mathbf{G}_W$ and $\mathbf{G}_{V_r}$ can be computed as follows

$$\begin{aligned} \mathbf{G}_W &= \sum_{l=1}^{L} \mathbf{G}_{W,l} = \sum_{l=1}^{L} \sum_{p=1}^{P_l} \mathbf{u}_{lp} \mathbf{y}'_{lp} = \mathbf{U}\mathbf{Y}', \\ \mathbf{G}_{V_r} &= \sum_{l=1}^{L} \mathbf{G}_{V_r,l} = \sum_{l=1}^{L} \sum_{p=1}^{P_l} \mathbf{x}_{l\xi(p,r)} \mathbf{y}'_{lp} = \mathbf{X}\mathbf{Y}', \end{aligned} \tag{9}$$

where $\xi(p,r)$ is the r-th vertex pointed by v_p, $\mathbf{G}_{W,l}$, $\mathbf{G}_{V_r,l}$ are the gradient contributions corresponding to E_l, that is to DOAG l. The gradient coordinates are denoted by $\mathbf{G}_{W,i,j,l}$ and $\mathbf{G}_{V_r,i,j,l}$, respectively. The *delta-error* y_{ilp} can be computed recursively according to BPTS:

$$y_{ils} = \begin{cases} m_i f'(a_{ils}) \beta'_+ (o_{ls} - d^+) f'(a_{ls}) & l \in C^+ \\ m_i f'(a_{ils}) \beta'_- (o_{ls} - d^-) f'(a_{ls}) & l \in C^- \end{cases} \tag{10}$$

$$y_{ilp} = f'(a_{ilp}) \sum_{r \in \mathcal{I}_p} \sum_{k=1}^{n} v_{ki\sigma(\mathcal{I}_p,r)} y_{klr}, \quad p \neq s. \tag{11}$$

These equations represent a vectorial form of *Back-Propagation Through Structure* (*BPTS*) gradient computational scheme [Sperduti and Starita, 1997].

[3]Note the difference between y_{ls} and $\mathbf{y}_{ls}$. While $y_{ls} \in \mathcal{R}$ is the delta error associated with the output unit, $\mathbf{y}_{ls} \in \mathcal{R}^n$ is the delta error corresponding to the supersource layer.

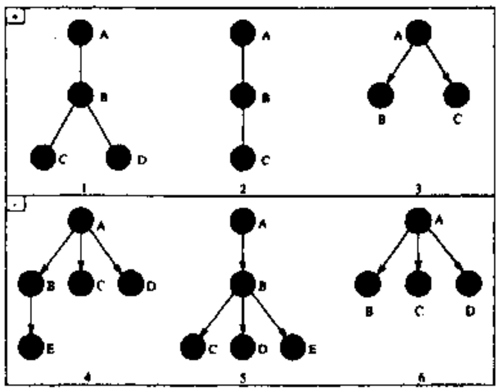

Figure 3: The problem of classifying DOAGs depending the their outdegree. Positive DOAGs are those with outdegree less than three. Since $\mathcal{D}_L \Uparrow = 8$, a network with 8 hidden units can be used in order to avoid local minima. Note that the letters attached to the nodes are only used as pointers.

The analysis proposed in this paper is based on the implicit assumption there exists at least one set of weights for which $E = 0$, that is we assume that all the DOAGs of the learning environment can be classified correctly.

3 A bound on the hidden units for efficient learning

In this section we study the role of graph topology regardless of the information attached to the nodes. This is harder than studying the case which there are also significant labels attached to the nodes, which are likely to simplify the learning process.

Definition 1 *Given DOAGs collection $\mathcal{D}_L$, let $\mathbf{U}_L \doteq \{\mathbf{u}_{lp} : p = 1, \ldots, P_l; \ l = 1, \ldots, L\}$ be the corresponding set of nodes obtained when using a topological sorting T_s. We define relation $\bowtie \subset \mathbf{U}_L \times \mathbf{U}_L$ as follows:*

$$\mathbf{u}_{hl} \bowtie \mathbf{u}_{km} \Longleftrightarrow DOAG(\mathbf{u}_{hl}) = DOAG(\mathbf{u}_{km}), \tag{12}$$

where $DOAG(\mathbf{u})$ denotes the DOAG having node $\mathbf{u}$ as a supersource.

Definition 2 *Given DOAGs collection $\mathcal{D}_L$, consider quotient set $\mathbf{U}_L/_{\bowtie}$. The number $\mathcal{D}_L \Uparrow \doteq |\mathbf{U}_L/_{\bowtie}|$ is referred to as the* power pointer *of $\mathcal{D}_L$.*

Theorem 1 *Given $\mathcal{E} \sim \{\mathcal{L}_e, \mathcal{N}, E(\cdot)\}$, assume that there exists at least a solution with $E = 0$. Then, function $E(\cdot)$ has no local minima different from $E = 0$, provided that $\mathcal{D}_L \Uparrow \leq n$.*

Proof: see the Appendix A. □ Note that $\mathcal{D}_L \Uparrow \leq n$ forces to use neural metworks with at least as many hidden units as DOAGs to learn, since $\mathcal{D}_L \Uparrow \geq L$. A special interesting case is that in which $\mathcal{D}_L \Uparrow \geq L$. This

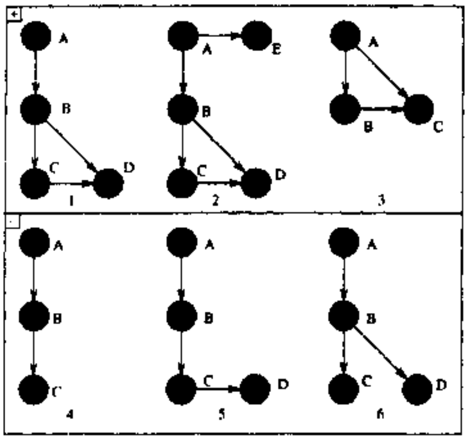

Figure 4: The problem of classifying DOAGs depending the presence of a common sub-graph. Positive DOAGs are those containing subgraph $\{(X,Y),(Y,Z),(X,Z)\}$. Since $\mathcal{D}_L \Uparrow = 9$, a network with 9 hidden units can be used in order to avoid local minima. Note that the letters attached to the nodes are only used as pointers.

holds for all those DOAGs with structure indicated in the following definition.

Definition 3 A set of DOAGs $\mathcal{D}_L$ is a *Chinese box* provided that $\forall\ \mathcal{U}_l \in \mathcal{S}_U,\ \ \exists u_l, \mathcal{U}_{l_1}, \ldots, \mathcal{U}_{l_k}\ :\ \ \mathcal{U}_l = \{(\mathbf{u}_l, \mathbf{u}_{l_1}), \ldots, (\mathbf{u}_l, \mathbf{u}_{l_k}), \mathcal{U}_{l_1}, \ldots, \mathcal{U}_{l_k}\}$.

Corollary 1 Let $\mathcal{D}_L$ be a *Chinese box*, with associated learning environment $\mathcal{L}_e$, and $\mathcal{N}$ a generalized recurrent network such that there exists at least a solution with $E = 0$. If $n \geq |\ \mathcal{L}_e\ | = L$ then error function $E(\cdot)$ has no local minima different from $E = 0$.

Proof: Trivial, since for Chinese boxes $\mathcal{D}_L \Uparrow =\ L$.

Example 1 Consider the problem of classifying DOAGs depending the their outdegree. Positive DOAGs are those with outdegree less than three (see Fig 3). It can can easily be checked that

$$
\begin{aligned}
\mathrm{U}_L/_{\bowtie} = {} & [(C-1, D-1, C-2, B-3, C-3, E-4,\\
& C-4, D-4, C-5, D-5, E-5, B-6, C-6, D-6),\\
& (B-1, A-3), (B-5, A-6), (B-2, B-4),\\
& (A-1), (A-2), (A-4), (A-5)].
\end{aligned}
$$

Hence, $\mathcal{D}_L \Uparrow =\ 8$. Because of Theorem 1, we conclude that when using a neural network with $n = 8$ hidden units, the given training set gives rise to an error function with no local minima.

Example 2 Consider problem of classifying DOAGs depending the presence of a common sub-graph and assume, in particular, that positive DOAGs are those containing subgraph $\{(A,B),(B,C),(A,C)\}$ (see Fig. 4). It can can easily be checked that

$$
\begin{aligned}
\mathrm{U}_L/_{\bowtie} = {} & [(D-1, D-2, E-2, C-3, C-4, D-5, C-6,\\
& D-6), (C-1, C-2, B-3, B-4, C-5),\\
& (B-1, B-2, A-3), (A-1), (A-2),\\
& (A-4, B-5), (A-5), (B-6), (A-6)].
\end{aligned}
$$

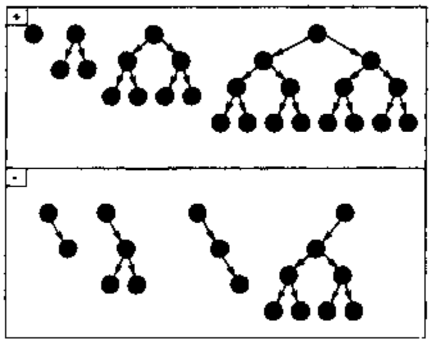

Figure 5: The problem of learning perfect binary trees. The associated error function has no local minima, provided that one uses 8 hidden units.

Hence, $\mathcal{D}_L \Uparrow = 9$, which means that can surely get rid of local minima when using neural networks with 9 hidden units.

Example 3 Consider the problem of learning the concept of *perfect binary tree* from the examples shown in Fig. 5. It can easily be proven that the given learning environment is a Chinese box and, therefore, according to Corollary 1, a network with 8 hidden units ensures that there are no local minima.

4 Conclusions

In this paper we have proven that if we use as many hidden units as the *pointer power* of the given collection of DOAGs, any problem of learning their classification can be solved with a unimodal error function. Moreover, in the special case of Chinese boxes, the *pointer power* reduces to the cardinality of the given training set. The given bound on the number of hidden units that are needed to avoid local minima, however, is not necessarily useful for the network design, since our condition is not related to generalization issues. The given bound, however, seems to be very interesting from a theoretical point of view, especially when compared with related results given for multilayer networks [Poston *et al.,* 1991; Yu, 1992].

Appendix

Proof: Sketch [4] - Theorem 1

According to the hypothesis, when using $\bowtie$, we can reduce $\mathcal{D}_L$ to $\mathcal{D}_L/_{\bowtie}$ and, because of the defined computation scheme, all $DOAG(\mathbf{u}_{l_p})$: $\mathbf{u}_{l_p} \in \mathbf{u}_{\bowtie}(\rho) \in$

[4] The basic idea of the proof is somewhat related to Poston *et.* al's [Poston *et a*l., 199l] and Yu's [Yu, 1992], for the case of multilayer perceptrons. L. Harney [Harney, 1994] pointed out that Poston *et al.* proof was not complete and that Yu's proof contained a flaw. Yu and Chen [Yu and Chen,.1995], however, have recently proven that the flaw pointed out by Harney can be fixed up and that the claimed result holds.

$\mathcal{D}_L/_{\bowtie}$ give rise to the same pointers on the network hidden layer. Let $\tilde{\mathbf{X}} \doteq [\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_{\mathcal{D}_L \Uparrow}] \in \mathcal{R}^{n, \mathcal{D}_L \Uparrow}$ be the matrix of different pointers and $D_\rho \doteq \{l : \mathbf{u}_{lp} \in \mathbf{u}_{\bowtie}(\rho)\}$. Equation (9) for the gradient w.r.t a generic matrix $\mathbf{V}_r$ can be re-organized to a sum with $\mathcal{D}_L \Uparrow$ different terms only, that is

$$\mathbf{G}_{V_r} = \mathbf{X}\mathbf{Y}' = \tilde{\mathbf{X}}\tilde{\mathbf{Y}}', \tag{13}$$

where

$$\tilde{\mathbf{y}}_{\rho p} = \sum_{l \in D_\rho} \mathbf{y}_{lp}, \tag{14}$$

and $\tilde{\mathbf{Y}} \doteq [\tilde{\mathbf{y}}_1, \ldots, \tilde{\mathbf{y}}_{\mathcal{D}_L \Uparrow}] \in \mathcal{R}^{n, \mathcal{D}_L \Uparrow}$. Since $n \geq \mathcal{D}_L \Uparrow$, matrix $\tilde{\mathbf{X}}$ is full rank with $P = 1$. Now, the proof can be given separately in the cases in which $rank\ \tilde{\mathbf{X}} = \mathcal{D}_L \Uparrow$ and $rank\ \tilde{\mathbf{X}} < \mathcal{D}_L \Uparrow$.

1. $rank\ \tilde{\mathbf{X}} = \mathcal{D}_L \Uparrow$.
   Under this condition, because of (13), the condition for critical points $\mathbf{G}_{V_r} = 0$ yields $\tilde{\mathbf{Y}} = 0$. Now we have two different cases for graphs in $\mathcal{D}_L$
   (a) $\mathcal{U}_l$: $\mathbf{u}_{ls} \in \mathbf{u}_{\bowtie}(\rho)$ and $\mid \mathbf{u}_{\bowtie}(\rho) \mid = 1$.
   In this case $\tilde{\mathbf{y}}_{ls} = 0 \implies \mathbf{y}_{ls} = 0$, which, because of BPTS equations (10) yields $E_l = 0$, in the case of $m_i \neq 0$ (if $m_i \neq 0$, see the *singular* case (2)).
   (b) $\mathcal{U}_l$: $\mathbf{u}_{ls} \in \mathbf{u}_{\bowtie}(\rho)$ and $\mid \mathbf{u}_{\bowtie}(\rho) \mid > 1$.
   Also in this case, we can prove that $\mathbf{y}_{ls} = 0$. Consider the generic condition that can be derived from (13), that is $\tilde{\mathbf{y}}_{\rho p} = \sum_{l \in D_\rho} \mathbf{y}_{lp} = 0$. We can remove all terms $\mathbf{y}_{ls}$ in (14), for which $\mathbf{y}_{ls} = 0$ were already proven in (1a) and, for the correspondent graphs we have $\mathbf{y}_{ls} = 0$, which, in turn, yields $E_l = 0$. Let $\mathcal{D}_F$ be the set of graphs whose terms in (14) were removed. All other graphs in $\mathcal{D}_L \setminus \mathcal{D}_F$ are necessarily subgraphs of elements in $\mathcal{D}_F$. For any $\mathcal{U}_l \in \mathcal{D}_L \setminus \mathcal{D}_F$ consider the corresponding encoding network of a graph $\mathcal{U}_\lambda$, such that $\mathcal{U}_l$ is a subgraph of $\mathcal{U}_\lambda \in \mathcal{D}_F$. According to the analysis carried in (1a) we derive that $\mathbf{y}_{\lambda s} = 0$. Since $\mathcal{U}_l$ is a subgraph of $\mathcal{U}_\lambda$, we can always find a node $\mathbf{u}_{p_m}$: $DOAG(\mathbf{u}_{\lambda, p_m}) = \mathcal{U}_l$. According to the BPTS equations (11), when following a path connecting s to p_m we derive that $\mathbf{y}_{\lambda, p_m} = 0$. If matrices $\mathcal{V}_r$ are full rank, then $\mathbf{y}_{ls} = 0$ and, finally, $E_l = 0$.
2. $rank\ \tilde{\mathbf{X}} < \mathcal{D}_L \Uparrow$
   Under this assumption, there is the possibility that $\forall r \quad \mathbf{G}_{V_r} = 0 \quad \Longleftarrow \tilde{\mathbf{Y}} \neq 0$. Now we prove that the corresponding configuration cannot be a local minimum for $E(\cdot)$. Assume, by contradiction, that configuration $\{\mathbf{W}_o, \overline{\mathbf{V}}_o, \mathbf{M}_o\}$ is a local minimum. Since $\tilde{\mathbf{X}} = n$ holds with $\mathcal{P} = 1$, there would be infinite global minima ($E = 0$) in the neighborhood of $\{\mathbf{W}_o, \overline{\mathbf{V}}_o, \mathbf{M}_o\}$, which is not consistent with the assumption that $E(\cdot)$ is a continuous function [5].

References

M.A. Arbib and Y. Given'on. Algebra automata i: Parallel programming as a prolegomena to the categorical approach. *Information and Control,* 12:331-345, 1968.

M.L. Brady, R. Raghavan, and J. Slawny. Back-propagation fails to separate where perceptrons succeed. *IEEE Transactions on Circuits and Systems,* 36:665-674, 1989.

M. Gori and A. Tesi. On the problem of local minima in backpropagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* PAMM4(1):76~86, January 1992.

L.G.C. Hamey. Comment on "can backpropagation error surface not have local minima?". *IEEE Transactions on Neural Networks,* 5(5):844, 1 September 1994.

M.L. Minsky and S.A. Papert. *Perceptrons - Expanded Edition.* MIT Press, Cambridge, 1988.

J. B. Pollack. Recursive distributed representations. *Artificial Intelligence,* 46(I-2):77-106, 1990.

T. Poston, C. Lee, Y. Choie, and Y. Kwon. Local minima and backpropagation. In *International Joint Conference on Neural Networks,* volume 2, pages 173-176, Seattle, (WA), July 1991. IEEE Press.

E.D. Sontag and H.J. Sussman. Backpropagation separates when perceptrons do. In *International Joint Conference on Neural Networks,* volume 1, pages 639-642, Washington DC, June 1989. IEEE Press.

A. Sperduti and T. Starita. Supervised neural networks for classification of structures. *IEEE Transactions on Neural Networks,* 8(3):714-735, 1997.

A Sperduti, A. Starita, and C. Goller. Learning distributed representations for the classification of terms. In *International Joint Conference on Artificial Intelligence,* pages 509-515, 1995.

X.H. Yu and G.A. Chen. On the local minima free condition of backpropagation learning. *IEEE Transactions on Neural Networks,* 6(5):1300-1303, September 1995.

X.H. Yu. Can backpropagation error surface not have local minima? *IEEE Transactions on Neural Networks,* 3(6):1019-1020, November 1992.

[5] Using related arguments we can prove that the assumptions that the pointer matrices be full rank and $m_i \neq 0$ can easily be removed.