ELSEVIER

Artificial Intelligence 132 (2001) 1–38

Artificial Intelligence

www.elsevier.com/locate/artint

Artificial nonmonotonic neural networks

B. Boutsinas [a,c,*], M.N. Vrahatis [b,c]

[a] *Department of Computer Engineering and Informatics, University of Patras, GR-26500 Patras, Greece*
[b] *Department of Mathematics, University of Patras, GR-26500 Patras, Greece*
[c] *University of Patras Artificial Intelligence Research Center (UPAIRC), University of Patras, GR-26500 Patras, Greece*

Received 30 August 1999; received in revised form 8 January 2001

Abstract

In this paper, we introduce Artificial Nonmonotonic Neural Networks (ANNNs), a kind of hybrid learning systems that are capable of nonmonotonic reasoning. Nonmonotonic reasoning plays an important role in the development of artificial intelligent systems that try to mimic common sense reasoning, as exhibited by humans. On the other hand, a hybrid learning system provides an explanation capability to trained Neural Networks through acquiring symbolic knowledge of a domain, refining it using a set of classified examples along with Connectionist learning techniques and, finally, extracting comprehensible symbolic information. Artificial Nonmonotonic Neural Networks acquire knowledge represented by a multiple inheritance scheme with exceptions, such as nonmonotonic inheritance networks, and then can extract the refined knowledge in the same scheme. The key idea is to use a special cell operation during training in order to preserve the symbolic meaning of the initial inheritance scheme. Methods for knowledge initialization, knowledge refinement and knowledge extraction are introduced. We, also, prove that these methods address perfectly the constraints imposed by nonmonotonicity. Finally, performance of ANNNs is compared to other well-known hybrid systems, through extensive empirical tests.

Keywords: Nonmonotonic reasoning; Neural networks; Hybrid systems; Inheritance networks; Unconstrained optimization; DNA sequence analysis

* Corresponding author.
E-mail address: vutsinas@ceid.upatras.gr (B. Boutsinas).

0004-3702/01/$ – see front matter
PII: S0004-3702(01)00126-6

1. Introduction

1.1. Motivation and background

Nonmonotonic reasoning plays an important role in the development of systems that try to mimic common sense reasoning, as exhibited by humans. Human beings are constantly forced to make decisions and reach conclusions in an ambiguous world. The knowledge that can be acquired by observation is inherently incomplete and may contain conflicting information as well as exceptions to general rules. Many formalisms are proposed in the literature that are capable of representing knowledge under a multiple inheritance scheme with exceptions (e.g., [3,6,9,19,26,48,50,54]). A nonmonotonic reasoner has to face the two general problems of knowledge-based systems, namely the strong dependency on the correctness of the domain knowledge and the lack of domain independent and effective learning algorithms. Due to the latter, the domain knowledge must be altered manually, whenever necessary. Moreover, a nonmonotonic reasoner, usually, has problems in dealing with multiple extensions of a theory. In such situations, extensions are treated as cases of ambiguity. The way they are treated depends on whether a credulous or a skeptical view is adopted [55]. Besides, no attempt is made to resolve possible conflicts (except for a few cases, as in [52]).

On the other hand, example-based systems, such as Artificial Neural Networks (ANNs), need a large set of training examples and they have a strong dependency on the features used to describe those examples. They also lack an explanation capability for the generated outputs and, consequently, for the decisions reached. Moreover, a longstanding problem in connectionist modelling is the representation of structured objects. Although some attempts have been made to address this problem (e.g., [47]), the proposed systems are not efficient [27]. On the contrary, an explanation capability and representation of structured objects can be easily provided by a knowledge-based system.

Recently, significant attention has been paid to the development of hybrid systems that are based on a neural-symbolic integration, aiming at exploring the advantages of each constituent paradigm. There were promising early attempts [11] toward the combination of the explanation capabilities and the powerful declarative knowledge representation of the symbolic approach with the massive parallelism and the generalization capabilities of the connectionist approach. Neural-symbolic hybrid systems use an Artificial Neural Network as an example-based learning system and a symbolic knowledge representation scheme for the domain knowledge. The most important contributions to this interesting field can be found in [1,8,10,12,18,22,29,30,42,56].

Following McCarthy's observation [37], most of the hybrid systems using neural networks for inferential processing, are based on logic-based propositional knowledge representation schemes. Such logic-based formalisms may lack several important properties of nonmonotonic reasoning [6], such as the very important property of *stability* (also called *cumulativity*). This is generally true even for well known nonmonotonic formalisms such as Reiter's *Default Logic* [6]. Special extensions of *Default Logic* have been introduced in order to tackle the stability problem, such as *Cumulative Default Logic* [4,34]. Moreover, it has been shown that many decision problems are intractable or even undecidable when stated within the context of formal logic-based systems.

On the other hand, symbolic-connectionist systems in a nonmonotonic domain are of great importance to both theory and practice. Path-based such formalisms are representation schemes based on semantic networks. They introduce an alternative approach to nonmonotonic reasoning. They try to tackle algorithmic intractability and the correct treatment of incomplete or contradictory knowledge. Although there exists a lot of criticism about the semantics of semantic networks [64], it is accepted that they can be effectively used as a representation and inference scheme in a nonmonotonic domain [54]. Inheritance networks is such a path-based formalism with widespread use in nonmonotonic reasoning systems [55].

Hybrid systems are usually based on logic-based knowledge representation schemes as far as the domain knowledge is concerned. The proposed, in this paper, Artificial Nonmonotonic Neural Networks (ANNNs) are hybrid systems that use inheritance networks, as a nonmonotonic multiple inheritance knowledge representation scheme for the domain knowledge, and Artificial Neural Networks as a learning mechanism. The latter are supported by a proper training method, which suits perfectly our approach and is applied by changing selected weights at each epoch. ANNNs are not based on energy minimization, but on the spreading activation metaphor. The input cells of the connectionist part are externally activated, based on known facts in the domain knowledge, and the spreading of this activation forces some output cells to be either activated or deactivated. It is the activation of output cells that guides the reasoning process.

1.2. Related work

In [23] a logic-based method is presented for inserting any propositional general logic program P into a three-layer feedforward Artificial Neural Network with binary threshold neurons. It is proved that if the network is transformed into a recurrent network, by connecting output units to corresponding input units, it always falls into a unique stable state that corresponds to the unique stable model, namely the semantics, of P. The potential impact of this result is that a new massively parallel computational model for logic programming is derived.

The system presented in [56] (Knowledge-Based Artificial Neural Network) is capable of inserting "if-then" rules into a neural network, refining these rules using a backpropagation based learning algorithm and extracting rules from the neural network. It is empirically shown that the system can not only revise the domain knowledge but also can efficiently learn new rules from examples using the domain knowledge.

Following the key idea in [23], the system in [12] (Connectionist Inductive Learning and Logic Programming System) integrates inductive learning from examples and domain knowledge with deductive learning from Logic Programming. Propositional logic programs can be inserted into a feedforward Artificial Neural Network with bipolar semi-linear threshold neurons. The network is trained using the standard backpropagation learning algorithm and the revised logic program can be extracted.

Moreover, it is shown [40–42] that the satisfiability problem in propositional logic can be reduced to the problem of finding a global minima of an energy function of a symmetric network. The consequence of this very important result is that symmetric neural networks

can be applied in solving a lot of hard problems, such as optimization problems and constraint satisfaction problems [16].

There are also efforts in the direction of hybrid systems that use knowledge representation schemes based on first-order logic. Unfortunately, such systems are still propositional [24,25]. In [21], an automated reasoning system for first-order Horn clauses is presented (Connectionist Horn Clause Logic), which is implemented in a feedforward neural network. In [24] an extension of the method in [23] is presented for inserting first-order logic programs into three-layer recurrent Artificial Neural Networks that correspond to an approximation of the semantics of programs. Finally, in [43] an analogous result to [42] is shown, according to which a first-order resolution proof (although of a fixed predetermined length) can be also reduced to a global minimum of the energy function of a symmetric network.

The first result toward a hybrid symbolic-connectionist system in a nonmonotonic domain is due to Pinkas. In [44], it is shown that the minimization problem of an energy function of a symmetric network can also be applied to propositional nonmonotonic reasoning (in fact, exceeding it [45]). As shown in [46], initial domain knowledge can be represented by a logic-based scheme, the *Penalty Logic*. To every propositional formula representing domain knowledge (the assumption), a positive real number is assigned (the penalty), in order to form a *penalty logic well formed formula*. This penalty has to be paid by any assignment that does not satisfy the assumption. An assignment is preferred over any other that pays a higher penalty. Assignments that pay the minimum penalty (preferred models) are preferred over any other. Pinkas showed that the minimization problem of an energy function of a symmetric network can be reduced to the problem of finding preferred models for a given set of assumptions representing initial domain knowledge.

In the rest of the paper, we first describe ANNNs. Then, we present a knowledge initialization, a knowledge refinement and a knowledge extraction method for ANNNs. Consequently, we test the performance of ANNNs against other well-known hybrid systems and, finally, we discuss some critical issues.

2. Artificial Nonmonotonic Neural Networks

Artificial Nonmonotonic Neural Networks are neural-symbolic hybrid systems. They possess a domain knowledge that is used for the initialization of an Artificial Neural Network. The latter is used as an example-based learning mechanism for the refinement of the initial knowledge through processing a set of classified examples. After the refinement, the acquired knowledge is extracted substituting the initial domain knowledge. At this state, a new cycle of knowledge initialization-refinement-extraction can start, whenever a new set of examples need to be considered (see Fig. 1).

Domain knowledge is represented by a nonmonotonic multiple inheritance scheme allowing exceptions. More specifically, we use Nonmonotonic Inheritance Networks (NINs) that are based on semantic networks. There is a clash of intuitions [55] in nonmonotonic inheritance networks concerning the treatment of nonmonotonicity (on-path versus off-path preemption), the treatment of competing extensions (forms of skepticism versus forms of credulity) and the direction in which the represented inheritance is followed

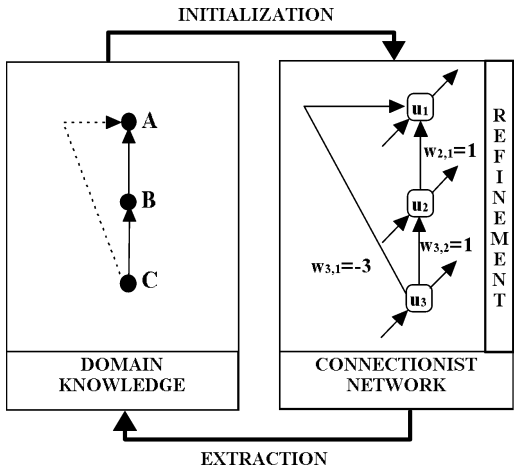

Fig. 1. The cycle of knowledge initialization, refinement and extraction.

(upward versus downward view of reasoning). Their performance is closely related to these intuitions [51]. The reasoning process was proved to be NP-hard for on-path/off-path credulous downward reasoners as well as for on-path upward credulous reasoners. However, it is proved to have polynomial time complexity for skeptical reasoners.

In NINs, knowledge is represented by attaching, to each node of a directed acyclic graph, a label that denotes an object, a class of objects or a property possessed by objects of the domain of discourse and by establishing the desired relationships through the insertion of the proper directed edges. When we insert an edge that emanates from a node and is incident to another, we mean that the class of objects represented by the former node inherits some or all of the defining properties of the class represented by the latter. If an inheritance exception exists, this is indicated by an exception link. For example, the NIN shown in the left part of Fig. 2 represents the facts that all Es are Ds (or are kinds of Ds, or are like Ds [3]), all Ds are Cs and so on. Moreover, a reasoning system can conclude that all Es are Cs because they are Ds. Finally, the exception link (indicated by a dotted line) represents that Ds are not Bs, although they are Cs.

There are two main operations associated with NINs. The first operation consists in answering whether an object possesses a particular property. The second consists in finding all the objects satisfying a particular set of properties. Developing efficient algorithms for these two operations is of great importance to many AI applications. Both of these operations can be reduced to computing the transitive relationships between the objects in the nonmonotonic inheritance network.

The objective of the knowledge initialization phase is to construct an Artificial Neural Network that provides the same answers as the nonmonotonic inheritance network, as far as the above operations are concerned. The knowledge initialization methodology we employ is presented in the next section. After the knowledge initialization phase, the constructed Artificial Neural Network is trained using a set of classified examples, in order to refine the initial knowledge. The refinement is achieved through changing the initial weights. The

way we construct the Artificial Neural Network during the initialization phase imposes some constraints on the training method we can use. We propose a new training method, which suits perfectly our approach, satisfying the requirements. The method is presented in Section 4. Finally, the refined knowledge is extracted from the Artificial Neural Network and it is represented by a nonmonotonic inheritance network, which replaces the initial one. The proposed knowledge extraction method is presented in Section 5.

3. The knowledge initialization method

The representation in the nonmonotonic inheritance network that corresponds to the domain knowledge can be based on either directed acyclic graphs or on set descriptions. The knowledge initialization phase is formally defined as follows:

Given: A directed acyclic graph $G = (V, E = R \cup POS \cup NEG)$ where V is the set of nodes that represent objects of the domain of discourse and E is the set of edges that represent relations between those objects. Set R consists of the edges that represent ordinary relations, set POS consists of the edges that represent exceptional positive relations and set NEG consists of the edges that represent exceptional negative relations. Finally, it is assumed that $R \cap POS \cap NEG = \emptyset$.

or:
(i) a set of relations $R = \{r \mid r = \langle q, s\rangle\}$, with some type of semantics (e.g., of the "isa", "ako" or "isl" [3] type), and
(ii) a set of exceptions $X = \{\langle q, \{n_1, \ldots, n_k\}, \{p_1, \ldots, p_l\}\rangle\}$, where $q, n_1, \ldots, n_k$, $p_1, \ldots, p_l$ represent objects of the domain of discourse and any n_i is in exceptional negative relation with q and any p_i is in exceptional positive relation with q.

initialize an Artificial Neural Network, with:

- a set of $k \in \mathbb{N}$ cells $U = \{u_1, \ldots, u_k\}$, each one of which receives a network binary input and gives a network binary output. Moreover, each cell u_i performs an operation S (to be described later) on its inputs;
- a set of integer weights $W = \{w_{i,j} \mid i, j \leqslant k\}$;
- a proper activation function σ applied to the network outputs:

$$\sigma(x) = \begin{cases} 1, & \text{if } x > 0; \\ -1, & \text{if } x \leqslant 0. \end{cases} \tag{1}$$

Cell input and activation are assumed to be discrete (see Section 4 for a justification).

In the following, we use interchangeably the notion of a link in the inheritance network and the notion of a weighted connection in the neural network. We first present the knowledge initialization method with respect to the main characteristics of common sense reasoning, namely nonmonotonicity, redundancy and ambiguity.

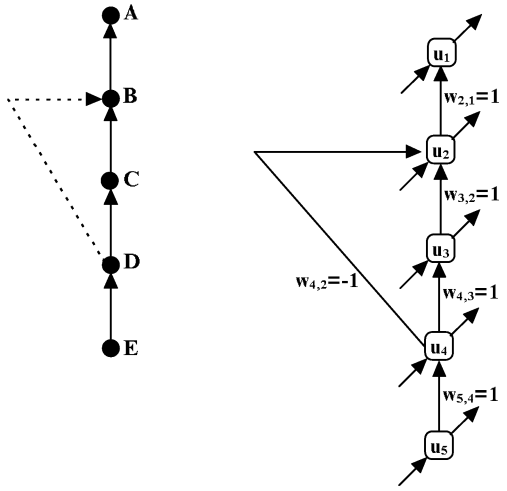

Fig. 2. An ANN initialized using a NIN.

3.1. Inserting symbolic knowledge with exceptions

Exceptions in the inheritance of properties, from more general classes to more specific ones, introduce nonmonotonicity in reasoning about the represented knowledge. The existence of more specific knowledge about an object of discourse may change previously valid conclusions.

Consider, for example, the nonmonotonic inheritance network shown in the left part of Fig. 2. An interpretation of the symbols could be the following: *A* stands for HOVER, *B* stands for FLY, *C* stands for BIRD, *D* stands for PENGUIN and *E* stands for MALE PENGUIN. The fact that an object of discourse is a bird leads to the conclusion that it flies. But the more specific fact that it is a penguin forces the conclusion that it does not fly.

The initial nonmonotonic inheritance network can be transformed to an Artificial Neural Network, as shown in the right part of Fig. 2. The equivalence between the ANN and the initial NIN, as far as its intended meaning is concerned, requires simulation of the inheritance cancelling (exceptions) represented by the negative links. This can be achieved by attaching a negative weight to the connection of the Artificial Neural Network that corresponds to the negative link of the symbolic domain knowledge, as shown in the right part of Fig. 2. In this way, the activation of cell u_4 or u_5 (corresponding to the fact that an object of discourse is a penguin or a male penguin) will prevent cells u_2 and u_1 from getting activated (corresponding to the conclusion that an object of discourse flies and hovers). The above are considered under a standard activation function and cell operation.

3.2. Inserting symbolic knowledge with redundancies

Since our aim is to simulate common sense reasoning as closely as possible, it is important for a nonmonotonic reasoner to handle *redundant information* in a consistent manner. Of course, as long as the reasoner's knowledge about the world remains unchanged

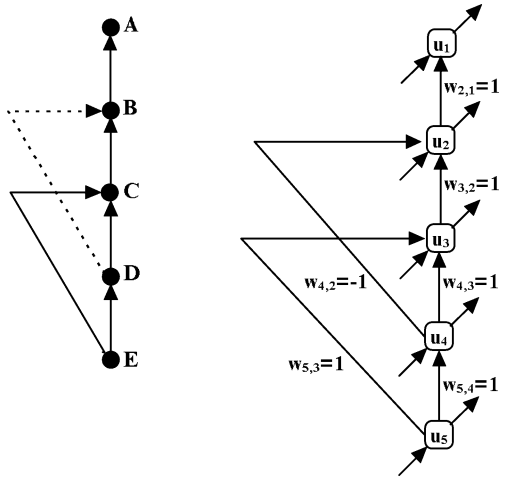

Fig. 3. A NIN supplemented with a redundant atomic statement.

in time, this kind of information could be recognized and, subsequently, removed during a preprocessing phase of the reasoner's life cycle. However, if the reasoner's knowledge is continuously subjected to revisions, as it is often the case, it is always possible that redundant, or even contradictory, information is introduced as a side effect of a revision process. Redundant information is closely related to the *stability* (*cumulativity*) of the reasoner.

Suppose that the reasoner, whose knowledge is represented by a nonmonotonic inheritance network, supports the conclusion $isa(X, Y)$. We say that the reasoner is stable if, after inserting into its knowledge base the (redundant) information that $isa(X, Y)$, it continues to support *all* the conclusions it supported before the insertion of this piece of information.

Consider, for example, the nonmonotonic inheritance network shown in the left part of Fig. 3, with the same interpretation of symbols as that of the previous example. Clearly, the redundant fact that a male penguin is a bird should not change any previous conclusion made without taking this redundant fact into consideration. This is satisfied by the activation of the cells in the right part of Fig. 3.

This kind of stability, is called *atomic stability*. It is obvious that an Artificial Neural Network initialized with the proposed methodology supports atomic stability. Atomic stability is viewed [26] as an acceptability criterion for an inheritance reasoner.

However, the other kind of stability, called *generic stability*, is not viewed as such a criterion. In our opinion, *both* atomic and generic stability should be viewed as acceptability criteria for an inheritance reasoner [2,3].

Consider, for example, the nonmonotonic inheritance network shown in the left part of Fig. 4, with the same interpretation of symbols as that of the previous example. Clearly, the redundant fact that a bird hovers should not change any previous conclusion that was made without taking this redundant fact into consideration. This is however not satisfied by the activation of the cells in the right part of Fig. 4. Without the connection from cell u_3

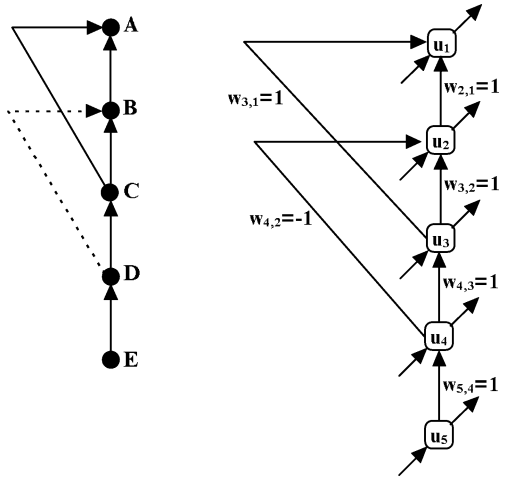

Fig. 4. A NIN supplemented with a redundant generic statement.

to cell u_1, activation of cell u_4 or u_5 prevents activation of cells u_2 and u_1. However, this is no longer true in the presence of that connection. Therefore, an Artificial Neural Network initialized with the methodology proposed so far does not support generic stability.

In order to overcome this serious problem, a more negative weight, say -2, could be attached to all the connections that correspond to the negative links. But, this technique does not always offer a solution, since this negative weight could be absorbed in the activation function of a standard type, if it is a function of the weighted sum of the inputs to a cell.

To address this problem, we define a cell operation S as the computation of the maximum absolute value of the inputs of the cell. Thus,

$$\forall u_i,\ S_i(in_1,\ldots,in_m) := in_1 \uplus in_2 \uplus \cdots \uplus in_m, \tag{2}$$

where $\uplus$ is a binary operator defined as:

$$x \uplus y = \begin{cases} -\frac{||x|-|y||+|x|+|y|}{2}, & \text{if } \frac{x-\varepsilon+y}{|x-\varepsilon+y|} < 0; \\ +\frac{||x|-|y||+|x|+|y|}{2}, & \text{if } \frac{x-\varepsilon+y}{|x-\varepsilon+y|} > 0 \ \wedge \ x, y > 0; \\ +1, & \text{otherwise.} \end{cases} \tag{3}$$

$in_1,\ldots,in_m$ are the inputs of cell u_i and $\forall u_j$ that is connected to u_i, in_j is derived by the function

$$in_j = \begin{cases} 0, & \text{if } \sigma(S_j) = -1; \\ w_{j,i} \uplus S_j, & \text{if } \sigma(S_j) = 1. \end{cases} \tag{4}$$

Intuitively, the cell operation S guarantees that the output of a cell is identical to its input with the maximum absolute value. In the case that there are more than one input of equal absolute value, the output is identical to the negative one. This is achieved by subtracting a very small positive number ε from some of the inputs determining the resolution of distinguishing x and y.

Therefore, (see Fig. 4) if $w_{4,2} = -2$ the output of cell u_2 is -2, if u_4 is activated. Eventually, the output of the cell u_2 becomes the output of cell u_1. Considering only the positive outputs as activating a cell, the Artificial Neural Network in Fig. 4 behaves correctly satisfying the very important property of *stability*.

3.3. Satisfying "*inferential distance ordering*"

It is often the case that a negative link should be preempted by a more specific positive one (or vice versa). Therefore, there is also an ordering of the exception links, either negative or positive, according to how specific is the class they refer to. The more specific the referring class the higher the priority of the exception link. This ordering is referred to, in the literature, as the "inferential distance ordering" [54] and it is viewed as an acceptability criterion for nonmonotonic reasoning. It ensures that if there are any conflicts among the exception links, they are resolved in favor of the exceptions that have a higher priority, according to "inferential distance ordering"; that is, in favor of the exceptions concerning more specific classes.

In order to properly initialize the Artificial Neural Network we have to preserve the "inferential distance ordering". But, if all negative links were represented by the same negative weight and all positive links by the same positive weight, we could not resolve the conflict in favor of the more specific one. We overcome this problem by assigning to a negative or a positive weight a negative or a positive link respectively, with a magnitude relative to a topological ordering of the cells.

According to a topological ordering, there is a function $f : U \to \mathbb{N}$ that assigns an integer to each cell such that, if cell u_i is an ancestor of cell u_j (in the sense that there exists a connection path from u_i to u_j), then $f(u_i) > f(u_j)$. We assign a weight to each exception link, starting from the links having tail nodes with the lowest topological order. If there are links having tail nodes with the same topological ordering, we start from a negative link(s). To each negative link, a negative weight, equal to the topological order of its tail node, is assigned. To each positive link, a positive weight, equal to the topological order of the tail node of the last examined negative link in the same path, is assigned. Exceptionally, to each positive link, in the case that its head node has a topological order lower than or equal to the head node of the last examined negative link (or of all the last examined negative links with the same tail node), a positive weight equal to the topological order of its tail node is assigned.

The above methodology, which is applied to the example of Fig. 5, is formally presented by the Initialization Algorithm in Section 3.5. Intuitively, this methodology guarantees that, along a path, every combination of positive and negative exception links satisfies *stability* and *inferential distance ordering* (see Section 5.4 for a proof).

3.4. Inserting symbolic knowledge with conflicts

Nonmonotonic multiple inheritance frequently introduces multiple extensions of a theory. We quote from [9]:

> "... an extension is a set of beliefs which are in some sense *justified* or *reasonable* in light of what is known about a world".

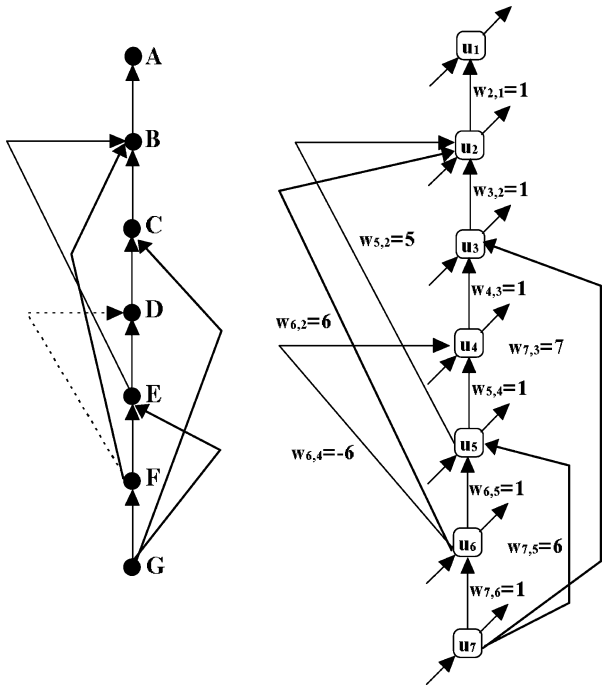

Fig. 5. Assigning negative or positive weights according to a topological ordering.

These multiple extensions are due to conflicts in the represented knowledge. Usually, all cases of multiple inheritance are treated as cases of ambiguity, e.g., in [54], and there is no effort to resolve the conflicts (an exception is [52]). The way in which multiple extensions are treated depends on whether a credulous or a skeptical view is adopted [26].

The proposed methodology, by definition, can not recognize multiple extensions and supports a preference over the conclusions represented by paths containing connections with weights of maximum absolute value. Consider, for example, the nonmonotonic inheritance network shown in the left part of Fig. 6. An interpretation of the symbols could be the following: A stands for PACIFIST, B stands for QUAKER, C stands for REPUBLICAN and D stands for NIXON. The activation of the cells in the right network of Fig. 6, supports the conclusion that, definitely, Ds are not As, since they are Cs, although the opposite conclusion is also supported, since As are Bs. This conclusion is due to the activation of cell u_3, that prevents u_1 from activation. Generally, any path that contains a connection with a maximum absolute value, due to the transfer of this value, is prevalent over any other. Therefore, if the weight is negative(positive) then a negative(positive) conclusion is derived.

One of the main advantages of the Artificial Nonmonotonic Neural Networks is that, during the knowledge refinement phase, explained in a later section, the conflicts can be resolved in favor of the most probable ones, with respect to the training set.

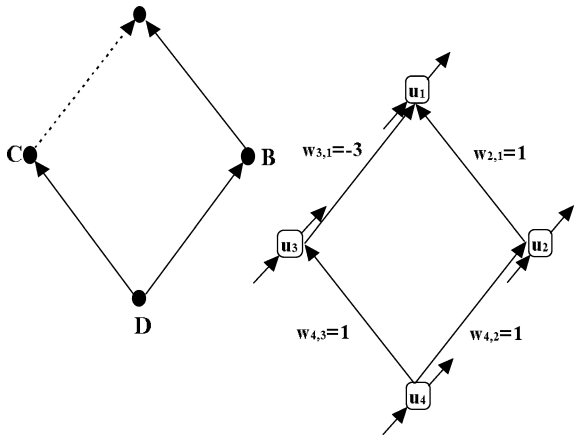

Fig. 6. Conflicted extensions.

3.4.1. Inserting symbolic coupled knowledge

In nonmonotonic reasoning, usually, objects are coupled with their immediate successors. This means that an object is not allowed to possess a property that is not possessed by any of its immediate successors, since this object possesses a property strictly because it belongs to a particular class. Of course, in some kind of nonmonotonic reasoning, the skeptical reasoning for instance, an object can not be coupled with its immediate successors.

In the proposed methodology, neither coupling nor decoupling can be assured. This is because the conclusion is relied on the maximum absolute value of the connections. Consider, for example, the nonmonotonic inheritance network shown in the left part of Fig. 7. Clearly, neither Ds nor Es are As, due to the negative weights $w_{3,2}$ and $w_{5,3}$. A skeptical reasoner, on the contrary, supports the conclusions that Es are As and that Ds are not As, thus allowing the decoupling of Es from Ds.

Of course, during the knowledge refinement phase, coupling or decoupling is preferred in favor of the most probable, with respect to the training set.

3.5. The initialization algorithm

According to the methodology presented in the above sections, the initialization algorithm is as follows:

Given: a direct acyclic graph: $G(V, E = R \cup POS \cup NEG)$,

or: a set of relations: $R = \{r \mid r = \langle q, s\rangle\}$
along with a set of exceptions: $X = \{\langle q, \{n_1, \ldots, n_k\}, \{p_1, \ldots, p_k\}\rangle\}$,

initialize an Artificial Neural Network with:

- a set of cells $U = \{u_1, \ldots, u_k\}$,
- a set of weights $W = \{w_{i,j} \mid i, j \leqslant k\}$,

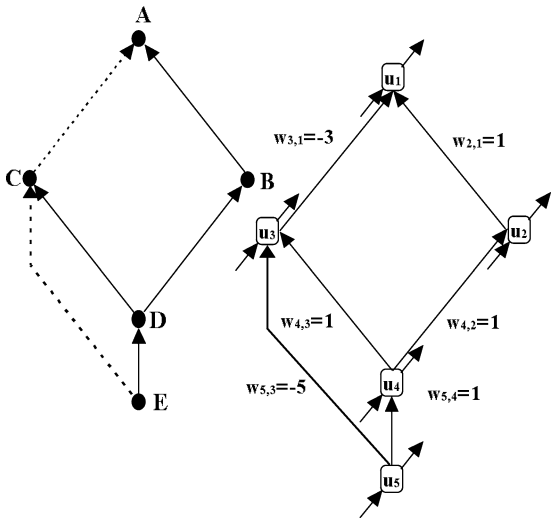

Fig. 7. Another example of conflicted extensions.

executing the following algorithm:

The initialization algorithm

1. for each $v_i \in V$ (or for each distinct q that is referred to in R), define a cell $u_i \in U$,
2. for each $e = (i, j) \in E$ (or for each $r = (i, j) \in R$) define a connection from u_i to u_j with $w_{i,j} = 1$,
3. for each $n = (i, j) \in NEG$ (or for each $\langle i, (j, \ldots), (\ldots)\rangle \in X$), in ascending topological order of i, define a connection from u_i to u_j with $w_{i,j} = -(d \times i)$ (where d is a positive integer that is used in order not to allow contiguous weights),
4. for each $p = (i, j) \in POS$ (or for each $\langle i, (\ldots), (j, \ldots)\rangle \in X$), in ascending topological order of i, define a connection from u_i to u_j with $w_{i,j} = +(d \times i)$ (where d is as above), if for the most closed, in the same path, edge $n = (x, y) \in NEG$ of p, $f(j) \geqslant f(y)$. Otherwise, define a connection from u_i to u_j with $w_{i,j} = +(d \times x)$.

4. A knowledge refinement method

Knowledge refinement is based on the training of the corresponding Artificial Neural Network. At the same time, knowledge refinement aims to an effective knowledge extraction. To this end, we need a training method that preserves the symbolic meaning of the initialized Artificial Neural Network. In our case, this is accomplished both by restricting the weights to be set to selected ones and by using a fixed architecture. Notice that most training algorithms are specialized to train a particular type of network architecture.

In our case, the network is considered as a network of neurons with discrete states. Thus, consider a Discrete Multilayer Neural Network (DMNN) consisting of L layers, in which the first layer denotes the input, the last one (L) is the output and the intermediate layers are the hidden layers. It is assumed that the $(l-1)$th layer has N_{l-1} units. These units operate according to the following equations:

$$net_j^l = \sum_{i=1}^{N_{l-1}} w_{ij}^{l-1,l} y_i^{l-1} + \theta_j^l, \quad y_j^l = \sigma^l\left(net_j^l\right), \tag{5}$$

where net_j^l is the net input to the jth unit at the lth layer, $w_{ij}^{l-1,l}$ is the connection weight from the ith unit at the $(l-1)$th layer to the jth unit at the lth layer, y_i^l denotes the output of the ith unit belonging to the lth layer, θ_j^l denotes the threshold of the jth unit at the lth layer, and σ is the activation function. We consider units where $\sigma(net_i^l)$ is a discrete activation function. We especially focus on units with two output states, usually called binary or hard-limiting units [38], i.e.:

$$\sigma^l\left(net_j^l\right) = \begin{cases} \text{“}true\text{”}, & \text{if } net_j^l \geqslant 0; \\ \text{“}false\text{”}, & \text{if } net_j^l < 0. \end{cases} \tag{6}$$

Although units with discrete activation function have been superseded to a large extent by the computationally more powerful units with analog activation function, DMNNs are still important in that they can handle many of the inherently binary tasks that neural networks are used for. Their internal representation is clearly interpretable, they are computationally simpler to understand than networks with sigmoid units and provide a starting point for the study of the neural network properties. Furthermore, when using hard-limiting units we can understand better the relationship between the size of the network and the training complexity [17]. In [13], it has been demonstrated that DMNNs with only one hidden layer can create any decision region that can be expressed as a finite union of polyhedral sets when there is one unit in the input layer. Moreover, artificially created examples were given, where these networks create non convex and disjoint decision regions. Finally, discrete activation functions facilitate neural network implementations in digital hardware and are much less costly to fabricate.

The most common feed forward neural network (FNN) training algorithm, back-propagation (BP) [49], which makes use of the gradient descent, cannot be directly applied to networks of units with discrete output states, since discrete activation functions (such as hardlimiters) are non-differentiable. This also holds for various modifications of the BP method (see, e.g., [31,32]). However, various modifications of the gradient descent approach have been presented in the literature [14,53,63]. In [15] an approximation to gradient descent, the so-called pseudo-gradient training method, is proposed. This method uses the gradient of a sigmoid as a heuristic hint instead of the true gradient. Experimental results validated the effectiveness of this approach.

We derive and apply a new training method for DMNNs that makes use of the gradient approximation introduced in [15]. Our method exploits the imprecise information regarding the error function and the approximated gradient, like the pseudo-gradient method does, but it has an improved convergence speed and has the potential to train

DMNNs in situations where, according to our experiments, the pseudo-gradient method fails to converge. For some comparative results of this method with BP see [33].

4.1. Problem formulation and proposed solution

We consider units with two discrete output states and use the convention f (or $-f$) for "false" and t (or $+t$) for "true", instead of the classical 0 and 1 (or -1, and $+1$). f, t are real positive numbers and $f < t$. Real positive values prevent units from saturating, give to the logic "false" some power of influence over the next layer of the DMNN and help the justification of the approximated gradient value, which we employ.

First, let us define the error for a discrete unit as follows:

$$e_j(t) = d_j(t) - y_j^L(t), \quad \text{for } j = 1, 2, \ldots, N_L, \tag{7}$$

where $d_j(t)$ is the desired response at the jth neuron of the output layer at the input pattern t, $y_j^L(t)$ is the output at the kth neuron of the output layer L. Notice that N refers to the number of output cells that are semantically related to the input cells. More specifically, these output cells are only those that eventually can be activated by the input cells. Thus, we consider a subgraph of the initial inheritance network. For a fixed, finite set of input–output cases, the square error over the training set, which contains T representative cases, is:

$$E = \sum_{t=1}^{T} E(t) = \sum_{t=1}^{T} \sum_{j=1}^{N_L} e_j^2(t). \tag{8}$$

The idea of the pseudo-gradient was first introduced in training discrete recurrent neural networks [65,66] and was extended to DMNNs [15]. The method approximates the true gradient $\nabla E(w)$ of the error function $E(w)$ with respect to the weights w, by introducing an analog set of values for the outputs of the hidden layer units and the output layer units.

Thus, it is assumed that y_j^l in Eq. (5) can be written as:

$$y_j^l = \widetilde{\sigma}^l\big(S\big(net_j^l\big)\big), \tag{9}$$

where, if $S(\cdot)$ is defined in $[0, 1]$ then:

$$\widetilde{\sigma}(x) = \begin{cases} \text{“}true\text{”}, & \text{if } x \geqslant 0.5; \\ \text{“}false\text{”}, & \text{if } x < 0.5, \end{cases} \tag{10}$$

otherwise if $S(\cdot)$ is defined in $[-1, 1]$ then

$$\widetilde{\sigma}(x) = \begin{cases} \text{“}true\text{”}, & \text{if } x \geqslant 0; \\ \text{“}false\text{”}, & \text{if } x < 0. \end{cases} \tag{11}$$

Using the chain rule, the pseudo-gradient is computed by:

$$\frac{\partial \widetilde{E}}{\partial w_{ij}^{l-1,l}} = \widetilde{\delta}_j^l y_i^{l-1}, \tag{12}$$

where the back-propagating error signal $\widetilde{\delta}$ for the output layer is:

$$\widetilde{\delta}_j^L = \big(d_j - S\big(net_j^L\big)\big) \cdot s'\big(net_j^L\big), \tag{13}$$

and for the hidden layers ($l \in [2, L-1]$) is:

$$\widetilde{\delta_j^l} = s'\big(net_j^l\big) \sum_n w_{jn}^{l,l+1} \widetilde{\delta_n^{l+1}}. \tag{14}$$

In these expressions $s'(net_j^l)$ is the derivative of the analog activation function.

By using real positive values for "true" and "false" we ensure that the pseudo-gradient will not reduce to zero when the output is "false". Notice also that we do not use σ', which is zero everywhere and nonexistent at zero. Instead, we use s', which is always positive, so $\widetilde{\delta_j^l}$ gives an indication of the direction and magnitude of a step up or down as a function of net_j^l in the error surface E.

However, as pointed out in [15], the value of the pseudo-gradient is not accurate enough, so gradient descent based training in DMNNs is considerably slow when compared to BP training in FNNs.

In order to alleviate this problem, we propose an alternative to the pseudo-gradient training method procedure. The proposed training method is applied by changing selected weights at each epoch, which is very useful in our approach. It is based on recently proposed unconstrained optimization methods [59–61].

Next, we present the proposed training method. For simplicity, we index the selected weights in sequence: $w^i,\ i = 1, 2, \ldots, n$. Thus, in order to find a point $w^* = (w_1^*, w_2^*, \ldots, w_n^*)$, which minimizes the given error function:

$$E : \mathcal{D} \subset \mathbb{R}^n \to \mathbb{R}, \tag{15}$$

in a specific bounded domain $\mathcal{D}$, we try to obtain a sequence of points $\{w^k\},\ k = 0, 1, \ldots$, which converges to w^*. So, using an arbitrary chosen starting vector of weights $w^0 = (w_1^0, w_2^0, \ldots, w_n^0) \in \mathcal{D}$, we subminimize E along the w_1 direction. Now, if $\widehat{w}_1$ is such a subminimizer, then, of course, point $(\widehat{w}_1, w_2^0, \ldots, w_n^0)$ possesses a smaller function value than point w^0. Then, we set $w_1^1 = \widehat{w}_1$. We repeat the above process to find a subminimizer $\widehat{w}_2$ along w_2 direction with as starting vector of weights $(w_1^1, w_2^0, \ldots, w_n^0)$. Thus, we compute the point w_2^1 and in the same way we compute the point w_3^1 with as starting vector of weights $(w_1^1, w_2^1, \ldots, w_n^0)$ and so on until point $w^1 = (w_1^1, w_2^1, \ldots, w_n^1)$ is formed. Now, after replacing the starting point w^0 by w^1, we can repeat the above process to compute w^2 and so on until the final estimated point w^* is computed according to a predetermined accuracy. Notice that we use only one-dimensional subminimization techniques to minimize the error function. For more details on such techniques see [7,36,39,57,58].

With the above discussion in mind we provide below a high level description of our algorithm, where E indicates the error function, $w^0 = (w_1^0, \ldots, w_n^0)$ the starting weights, $h = (h_1, \ldots, h_n)$ the starting stepsizes in each coordinate direction, *MEP* the maximum number of epochs required and δ, ε the predetermined desired accuracy.

The training algorithm

1. **Input** $\{E; w^0; h; MEP; \delta; \varepsilon\}$.
2. **Set** $k = -1$.
3. **Set** $i = 0$.

4. **If** $k < MEP$, replace k by $k+1$ and go to the next step;
   otherwise, go to Step 12.
5. **Replace** i by $i+1$ and continue.
6. **Compute** a subminimizer $\widehat{w}_i$, within an accuracy δ, along the ith direction by applying any one-dimensional subminimization iterative scheme.
7. **If** $\widehat{w}_i$ is a subminimizer of E along the ith direction set $w_i^{k+1} = \widehat{w}_i$;
   otherwise set $w_i^{k+1} = w_i^k$.
8. **If** $i < n$, go to Step 5.
9. **If** $E(w^{k+1}) \leqslant E(w^k)$, go to Step 3;
   otherwise set $y^0 = w^k$ and continue.
10. **Apply** one step of a pseudo-gradient training method utilizing the starting value y^0 and take its output value y^{SUB}.
11. **If** $E(w^{k+1}) > E(y^{SUB})$, then set $w^{k+1} = y^{SUB}$ and return to Step 3.
12. **Output** $\{w^k; E(w^k)\}$.

Our experience is that in many cases, as well as for all the problems studied in [33, 60,61], the application of the subprocedure at Step 10 is not necessary. We have placed it in our algorithm for the sake of completeness. For a proof of the convergence of this algorithm and related ones, see [59–61].

5. The knowledge extraction method

One of the main advantages of a hybrid system comes from the fact that the symbolic part is used to access the refined knowledge. Therefore, the knowledge extraction phase, where the refinement knowledge provided by the trained Artificial Neural Network is extracted in a comprehensible symbolic scheme, is of great importance to the reliability of a hybrid system.

The proposed knowledge extraction method of Artificial Nonmonotonic Neural Networks heavily relies on reversing the initialization phase. The cells of the Artificial Neural Network are, simply, transformed into nodes of the Nonmonotonic Inheritance Network. But the refinement of the initialized knowledge is actually represented by the changes in the weights of the connections of the Artificial Neural Network. These changes actually realize the refinement of the initialized knowledge, resolving conflicts. The initial knowledge of the inheritance network is changed due to insertions and deletions of exception links.

As it is well known, there is not a unique trained set of weights w^* that verifies the corresponding error function. To ensure that the changes in the weights do not concern connections that do not contribute in the knowledge refinement, we apply the training algorithm to selected weights, as described in Section 5.3.

5.1. Resolving conflicts

In resolving conflicts, a technique based on the identification of the extension that is supported by the set of classified examples used during the refinement phase is employed. Extensions are actually represented by paths of the inheritance network, and, hence,

of the connectionist network. The identification of the prevalent extensions, after the refinement phase, is achieved by the identification of changes in the weights attached to the connections.

In the initialized connectionist network prevalent extensions are those represented by a path containing connections with weights of maximum absolute value. Since negative weights support negations, a decrease of a weight gives a further precedence to a prevalent extension supporting negations. On the other hand, an increase in a weight, contributes toward the cancelling of a negative weight and therefore gives precedence to a prevalent extension supporting a positive result. An increase or a decrease in a weight, $w_{i,j}$, is defined with regard to its value before the refinement phase, $w^0_{i,j}$. When a prevalent extension is identified by a change in a weight, the extracted inheritance network is modified in order to support this extension. This is achieved by adding to the inheritance network a proper exception link, positive or negative, depending on the result supported by the identified extension. The added exception link concerns the extension as a whole and, clearly, it does not concern the exception link whose corresponding connection has changed. Therefore, the added exception link is attached to the path that represents the extension.

Consider, for example, the trained Artificial Neural Network of Fig. 8. Suppose that there is a decrease in the weight $w_{3,2} = -6$, ($w^0_{3,2} = -3$), which identifies the extension $F \to D \to C \to A$ as prevalent. Then, a negative exception link (F, A) is added that supports this prevalent extension. Notice that the added link does not affect the existed negative exception link (C, A), which has to remain. The added link, of course, has a priority over the existing one, according to the "inferential distance measure".

In order to add exception links properly, we need to identify the path that represents a certain extension. Since a prevalent extension is identified by a change in a weight of a connection that represents an existing exception link, we assume that an extension is

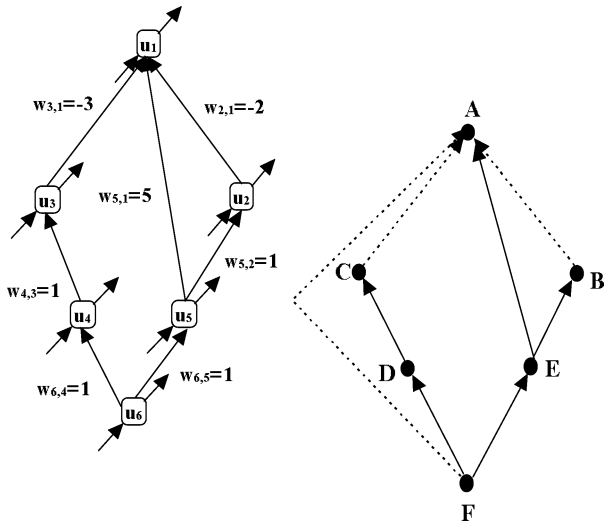

Fig. 8. Adding exception links.

represented by a path containing the tail and the head nodes of this exception link. Such a path is the $F \to D \to C \to A$ in Fig. 8, containing the tail node F and the head node A of the existing exception link (C, A).

In general, if there exists a path from a node representing an input cell under consideration to a node representing an output node under consideration and this path contains the changed weight, we add an exception link between these two nodes. The exception link is positive (negative) if there is an increase (decrease) to a negative (positive) weight or a decrease (increase) to a positive (negative) weight.

5.2. Preemption of exception links

It may also be the case that an existing exception link should be preempted, because it is not supported by the set of training examples. Such exception edges can be also identified by changes in weights of the corresponding connections.

Consider, for example, the trained Artificial Neural Network of Fig. 9 that was initialized as shown in Fig. 5. Suppose that weight $w_{6,4}$ is decreased. Therefore a positive exception link (F, D) is added. Notice that the added link introduces a conflict. This conflict is resolved in the implementation level, where the existing exception link (F, D) is deleted (see the extraction algorithm in the next subsection).

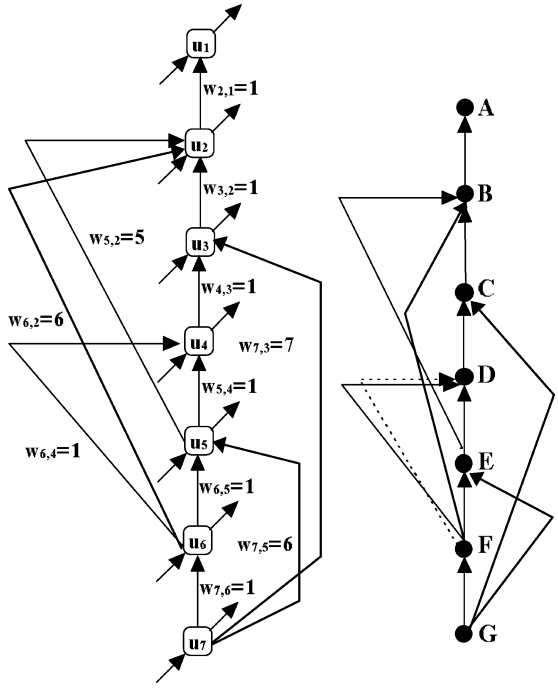

Fig. 9. Preempted exception links.

5.3. The extraction algorithm

To attack all the above cases, during the refinement phase, the training method is applied on selected weights. Mainly, these weights are the ones that correspond to exception links. Therefore, any change to a weight guarantees the prevalence or the validation of an exception link. Meanwhile, when resolving conflicts, if there are extensions represented by paths without exceptions, then identification of the prevalent extension is not possible, because only the weights that correspond to exception links are considered. Actually, in such a case the initialized connectionist network can not be trained.

We handle this problem by allowing the addition of weights to the set of selected weights that correspond to ordinary links. These are the weights of the connections that are adjacent to those output cells that represent nodes without incoming exception links.

Consider, for example, the trained Artificial Neural Network of Fig. 11. The training algorithm allows the addition of the weight $w_{563,561}$ to the set of selected ones, apart from $w_{560,559}$ and $w_{561,560}$.

The extraction algorithm, at first, transforms cells and connections to nodes and edges. Then, the algorithm adds proper negative or positive exception links, if a prevalent extension is identified. The extraction algorithm is presented below:

Given: the trained connectionist network with:
a set of cells $U = \{u_1, \ldots, u_k\}$, a set of weights $W = \{w_{i,j} \mid i, j \leqslant k\}$ along with the initial set of weights $W^0 = \{w^0_{i,j} \mid i, j \leqslant k\}$ before the refinement phase,

Revise the Inheritance Network with:

- a set of nodes V;
- a set of edges that represent ordinary relations R;
- a set of edges that represent exceptional positive relations POS;
- a set of edges that represent exceptional negative relations NEG.

Executing the following algorithm:

The extraction algorithm

1. **for** each cell $u_i \in U$:
   construct a node $v_i \in V$
2. **for** each connection from u_i to u_j:
   if $w_{i,j} = 1$ **then**
   construct an edge $e = (i, j) \in R$
   if $w_{i,j} < 0$ **then**
   construct an edge $e = (i, j) \in NEG$
   if $w_{i,j} > 0$ **then**
   construct an edge $e = (i, j) \in POS$
3. **for** each connection from u_i to u_j
   where $|w^0_{i,j}| \neq |w_{i,j}|$:
   if a path h, t containing (i, j) exists,
   where h is the node representing input under consideration
   and t is the node representing output under consideration

then
if $w^0_{i,j} < w_{i,j}$ **then**
add (if it does not already exist)
the exception link $e = (h, t) \in POS$.
delete any existing exception link $e = (h, t) \in NEG$.
elseif $w^0_{i,j} > w_{i,j}$ **then**
add (if it does not already exist)
the exception link $e = (h, t) \in NEG$.
delete any existing exception link $e = (h, t) \in POS$.

5.4. Soundness and completeness

The proposed methodology is sound and complete, as it is proved in the following. In general, we consider that:

a system is *sound* if every output is valid,
a system is *complete* if every valid output can be produced.

The proof of soundness and completeness of ANNNs is actually reduced to the proof of soundness and completeness of initialization, revision and extraction phase separately.

The completeness of the initialization phase is obvious and comes straightforward from the initialization algorithm in Section 3.5. Every inheritance network can be transformed to a neural network of the type described throughout Section 3. This is so, because there is a one to one correspondence between the sets of nodes V and edges E of the inheritance network and the sets of cells U and connections W of the neural network respectively. Obviously, there is always a topological ordering of nodes in the inheritance network, therefore it is always possible to assign weights to the connections of the neural network.

The soundness of the initialization phase guarantees the equivalence between the neural network and the intended meaning of the background knowledge. We consider that equivalence is guaranteed if the neural network satisfies *inheritance*, *stability* and *inferential distance ordering* (described in detail in Section 3). We prove the soundness of the initialization phase by cases. Lemmas 1 and 2 guarantee the satisfaction of the inheritance property. Satisfaction of the inferential distance metric and the stability property is guaranteed by Theorems 3 and 6. Notice that the conflict resolution property is guaranteed after the refinement phase, therefore all the following proofs refer to an intra-extension (intra-path) domain.

Lemma 1. *In a neural network constructed from an inheritance network without any exceptions or only positive ones, using the algorithm in Section* 3.5, *a cell* $o \in U$ *is activated iff at least one cell* $jk \in U$ *connected to* o *via a path of connections* $p = \{w_{j0,j1}, w_{j1,j2}, \ldots, w_{jn,o}\}$, *where* $j0, \ldots, jn \in U$ *and* $w_{j0,j1}, w_{j1,j2}, \ldots, w_{jn,o} \in W$, *is also activated.*

Proof. Suppose that cell jk, where $1 \leqslant k \leqslant n$, is activated. Thus, from Eq. (2) we obtain $S_{jk+1}(\ldots, in_{jk}, \ldots) = \cdots \uplus in_{jk} \uplus \cdots$ and, since jk is activated, from Eq. (4) we obtain

$\sigma(S_{jk}) = 1$ and $in_{jk} = w_{jk,jk+1} \uplus S_{jk}$. Given that there exist only positive exceptions, $w_{jk,jk+1} \geqslant 1$, $S_{jk+1} \geqslant 1$ and $\sigma(S_{jk+1}) = 1$. Similarly, $\sigma(S_{jk+2}) = 1$ and so on, until, finally, $\sigma(S_o) = 1$. □

Lemma 2. *In a neural network constructed from an inheritance network with at most one negative exception in each of its paths and no positive ones, using the algorithm in Section* 3.5*, the activation of cell $t \in U$, which the exception link starts from, prevents from activation any other cell connected to the head $h \in U$ of the exception link via a path of connections $p = \{w_{h,j1}, w_{j1,j2}, \ldots\}$, where $j1, j2, \ldots \in U$ and $w_{h,j1}, w_{j1,j2}, \ldots \in W$.*

Proof. Since cell t is activated and there exists only one negative exception, $w_{t,h}$, $S_t(\cdots) \geqslant 1$ and $\sigma(S_t) = 1$. Thus, from Eq. (2) we obtain $S_h(\ldots, in_t, \ldots) = \cdots \uplus in_t \uplus \cdots$ and by means of Eq. (4) we have $in_t = w_{t,h} \uplus S_t$. Because there exists only one negative exception, $w_{t,h} < -1$, since $f(t) < f(h)$ and $f(h) \leqslant 1$. Thus, $in_t < -1$, hence $S_h < -1$ and $\sigma(S_h) = -1$. Similarly, by Lemma 1, for any jk included in p, $\sigma(S_{jk}) = -1$. □

Theorem 3. *In a neural network constructed from an inheritance network with at most two exceptions $w_{ft,fh}, w_{st,sh}$ in each of its paths $p = \{w_{j0,j1}, w_{j1,j2}, \ldots, w_{jn,o}\}$, where $j0, \ldots, jn \in U$ and $w_{j0,j1}, w_{j1,j2}, \ldots \in W$, the inferential distance metric and stability property are satisfied. The following cases exist*:

(1) *Both of them are negative or positive. In this case the following sub cases are distinguished*:
   (a) *exclusion*: $f(ft) < f(fh) \leqslant f(st) < f(sh)$;
   (b) *inclusion*: $f(ft) \leqslant f(st) < f(sh) \leqslant f(fh)$;
   (c) *intersection*: $f(ft) \leqslant f(st) < f(fh) < f(sh)$.

(2) *The first is positive and the other is negative. In this case the following sub cases are distinguished*:
   (a) *exclusion*: $f(ft) < f(fh) \leqslant f(st) < f(sh)$;
   (b) *inclusion*: $f(ft) \leqslant f(st) < f(sh) \leqslant f(fh)$;
   (c) *intersection*: $f(ft) \leqslant f(st) < f(fh) < f(sh)$. *This subcase, actually, introduces atomic stability.*

(3) *The first is negative and the other is positive. In this case the following sub cases are distinguished*:
   (a) *exclusion*: $f(ft) < f(fh) \leqslant f(st) < f(sh)$;
   (b) *inclusion*: $f(ft) \leqslant f(st) < f(sh) \leqslant f(fh)$;
   (c) *intersection*: $f(ft) \leqslant f(st) < f(fh) < f(sh)$. *This subcase, actually, introduces generic stability.*

Proof. It is straightforward from Eqs. (2), (3) and (4) that for each of the above subcases, considering arbitrary input and output cells, the output cells are properly activated/deactivated so that the inferential distance metric and stability property are satisfied. In the following, we prove this claim for one of the above subcases. The proofs for the rest subcases are similar. Consider the case ((ii)(a)) above. Suppose that cell ft is activated. Then, from Lemma 1, for every node fi for which $f(ft) < f(fi) < f(sh)$, $\sigma(S_{fi}) = 1$ is implied. It is clear, from the fourth step of the initialization algorithm in

Section 3.5, that $|w_{ft,fh}| = |w_{st,sh}|$, $w_{ft,fh} > 0$, $w_{st,sh} < 0$. Therefore, for the output cell sh, $S_{sh}(in_{sm}, in_{st}) = in_{sm} \uplus in_{st}$, where sm is the immediate ancestor of sh. Then, $in_{sm} = w_{sm,sh} \uplus S_{sm} = w_{ft,fh}$. Also, $in_{st} = w_{st,sh} \uplus S_{st} = w_{st,sh}$ due to ε in Eq. (3). Therefore, $S_{sh} = w_{st,sh}$, also due to ε in Eq. (3). Finally, $\sigma(S_{sh}) = -1$. □

Definition 4. If $m = (mt, mh)$, $n = (nt, nh)$ are exception links, we define the *dominant* exception link as the link that has priority over the other, according to the inferential distance metric and the stability property with respect to Theorem 3. Formally, m is dominant on n, denoted as $m \succ n$, if

$$\begin{cases} sign(S_{nh}) = sign(w_m), & \text{if } sign(w_m) \neq sign(w_n); \\ f(mt) \geqslant f(nt),\ f(mh) \geqslant f(nh), & \text{if } sign(w_m) = sign(w_n). \end{cases} \tag{16}$$

Lemma 5. *The dominant operator, $\succ$, defines a relation D on the set of connections W. Relation D is reflexive and antisymmetric.*

Proof. Obviously, $\forall m = (mt, mh) \in W, m \succ m$, since $f(mt) = f(mt)$. Therefore D is reflexive. Obviously, $\forall m = (mt, mh), n = (nt, nh) \in W$, for which $m \succ n, n \succ m$, then $m \equiv n$ is implied. Therefore D is antisymmetric. □

Theorem 6. *The general case of a neural network constructed from any inheritance network is reduced to the case of Theorem* 3*, where at most two exceptions in each of its paths are allowed. This is accomplished by successively replacing the pairs of exceptions (m, n) by the dominant exception of them, starting from the pair with the lower topological order for the tail of m. Under this constraint, we prove that D is also transitive and hence permits successive substitutions.*

Proof. We prove that $\forall l = (lt, lh), m = (mt, mh), n = (nt, nh) \in W$, for which $f(lt) > f(lh)$, $f(mt) > f(mh)$, $f(nt) > f(nh)$, $f(lt) \geqslant f(mt) \geqslant f(lh) \geqslant f(mh)$, $f(mt) \geqslant f(nt) \geqslant f(mh) \geqslant f(nh)$, if $l \succ m, m \succ n$ then $l \succ n$ is implied. Hence, D is transitive. If $sign(w_l) = sign(w_m)$ (both negative or positive), from Definition 4, $f(lt) \geqslant f(mt)$, $f(lh) \geqslant f(mh)$. If, also, $sign(w_m) = sign(w_n)$ then $sign(w_l) = sign(w_m) = sign(w_n)$ and from Definition 4 $f(mt) \geqslant f(nt)$, $f(mh) \geqslant f(nh)$. So, $f(lt) \geqslant f(nt)$, $f(lh) \geqslant f(nh)$ and, from the second case of Definition 4, $l \succ n$. Else, if $sign(w_m) \neq sign(w_n)$ then, also, $sign(w_l) \neq sign(w_n)$ and, since $m \succ n$, $sign(S_{nh}) = sign(w_m)$. Therefore, $sign(S_{nh}) = sign(w_l)$ and, hence, from the first case of Definition 4, $l \succ n$. Consider, now, the case where $sign(w_l) \neq sign(w_m)$. If, also, $sign(w_m) \neq sign(w_n)$ then $sign(w_l) = sign(w_n)$. Since, by constraint, $f(lt) \geqslant f(nt)$, $f(lh) \geqslant f(nh)$ then, from the second case of Definition 4, $l \succ n$. Finally, if $sign(w_m) = sign(w_n)$ then $sign(w_l) \neq sign(w_n)$. Since $l \succ m$ then $sign(S_{mh}) = sign(w_l)$. Moreover, $sign(S_{nh})$ is determined by $sign(w_m)$ and $sign(S_{mh})$. But, since $sign(S_{mh}) = sign(w_l)$ then $sign(S_{nh})$ is determined by the dominant of l and m, which is l. Therefore, $sign(S_{nh}) = sign(w_l)$, and, from the first case of Definition 4, $l \succ n$. □

Completeness and soundness of the refinement phase is reduced to convergence of the training algorithm. For a proof of convergence of this algorithm as well as for some other related results see [59–61].

Completeness of the extraction phase is obvious and comes straightforward from the extraction algorithm in Section 5.3. The extraction algorithm guarantees that every output cell $o \in U$ can be reached by an input cell $i \in U$ by at least one path $p = \{w_{i,j1}, w_{j1,j2}, \ldots, w_{jn,o}\}$, where $j1, \ldots, jn \in U$ and $w_{i,j1}, \ldots, w_{jn,o} \in W$, which includes a weight $w_{p,q}$ that belongs to the set of the selected weights. Therefore, every change to a weight that belongs to the set of the selected weights, during the refinement phase, can be transformed to an addition of an exception link between the head and the tail of the path(s) which the changed weight is assigned to.

Soundness of the extraction phase is, also, obvious. An increase in a negative weight or a decrease in a positive weight actually decreases the absolute value of the weight and hence gives precedence to any other conflicted path with a positive or negative weight respectively. Similarly, a decrease in a negative weight or an increase in a positive weight gives precedence to the path that contains it.

6. Experimental results

In order to test the effectiveness and efficiency of ANNNs, as a learning system, we need to refine some initial knowledge of a pure nonmonotonic domain. Instead of choosing such a scarcely mentioned in the literature domain, which would not be appropriate for comparative results, we tested ANNNs in a problem that can be considered as being defined in a nonmonotonic domain, although in the literature, it is not treated as such. The problem is the extraction of classification rules from a dataset that can be considered, in the scope of this work, as being reduced to the extraction of general patterns and their exceptions. Classification problem, in a monotonic domain, has been attacked using both symbolic and connectionist techniques. There are also hybrid systems, as most of those mentioned in the introduction, that have been evaluated using it.

The key idea behind using ANNNs in the classification problem, under a pseudo nonmonotonic domain, is to consider some initial, arbitrarily chosen, classification rules that introduce conflicts and then to refine them by resolving these conflicts. Therefore, exceptions in the initial knowledge are, actually, artificially defined. The reader should bear in mind that ANNNs are capable of refining initial knowledge of a nonmonotonic domain. To our knowledge, there exists no hybrid system, apart from Pinkas's symmetric networks based on a translation of Penalty Logic, (which however heavily relies on user defined penalties, which in turn require a kind of preprocessing of the conflicts), that can be used in classifying examples that belong to a nonmonotonic domain. Of course, a relation can be established between Logic Programming and nonmonotonic reasoning through default and autoepistemic logics, which is based on treating negation [35]. Thus, hybrid systems like those in [23] and [12] can be, also, used in a nonmonotonic domain. However, for practical problems, one should take into consideration that, in establishing a relation between Logic Programming and nonmonotonic reasoning, the default interpretation of negation is an NP-complete task. Therefore, heuristics as "negation as failure to prove" has to be adopted. Moreover, one should also take into consideration hurdles imposed by logic-based formalisms, in general, such as the lack of the *stability* property, mentioned in the introduction.

In order to evaluate ANNNs, by obtaining comparative results with other hybrid systems, we transform a monotonic domain to an artificial nonmonotonic. It is actually this transformation that prevents ANNNs from outperforming alternative systems in classification in a monotonic domain. The main problem is that ANNNs, inherently, treat noise as exceptions and thus tend to overspecialize. Therefore, ANNNs are not so accurate in unseen examples, when applied to a monotonic domain. But, the obtained experimental results, as presented in what follows, are comparable to other hybrid systems and superior to various well-known pure symbolic or connectionist systems.

In the next subsection, we compare ANNNs with pure symbolic and connectionist systems against the problem of classifying both real-world datasets and test datasets. More specifically, we evaluate ANNNs using a real-world dataset representing the customer base of a big telecommunications company and two real-world datasets from the domain of Molecular Biology, especially that of DNA sequence analysis, namely the "*promoter recognition*" and the "*splice-junction determination*" problems [62]. Notice that DNA sequence analysis problems are used as benchmarks for comparing the performance of learning systems.

6.1. Extracting customer profiles

Classification rules can be extracted using supervised learning methods and can be used to classify data into predefined classes, described by a set of concepts (attributes). In most of the cases, independently of the adopted representation scheme, a set of classification rules describes a class through defining a general pattern with exceptions. A subset of these rules defines the general pattern (e.g., "young loan applicants are of a high risk"), while the rest define the exceptions (e.g., "young loan applicants with high income are of low risk").

Consider, for example the sample of rules shown in Fig. 10. These rules are constructed by the CN2 algorithm [5] and describe the behavior of the customer base of a big telecommunications company.[1] The conditions of the rules are certain attributes of the customer base and the predefined classes denote a profit related behavior. It is obvious that the last three rules define general patterns, while the first one is an exception to the last two rules.

Therefore, if we represent general patterns as an initial knowledge using a nonmonotonic inheritance scheme, we can find their exceptions refining the initial knowledge with respect to database records as training examples. Initial knowledge is, usually, provided by experts. In the tests we performed for ANNNs evaluation, it is chosen randomly. If we consider the three general patterns shown in Fig. 10 as initial knowledge, the initial inheritance network shown in Fig. 11 is constructed.

Consider the following interpretation[2] of the symbols:

- A stands for BAD;
- B stands for COMMON;
- C stands for GOOD;

[1] For confidentiality reasons, the classes in the rules are not the same as the actual ones.

[2] For confidentiality reasons, attribute values are not the same as the actual ones.

Young people, employees of government companies, residents of Aegean Islands are common customers
Entropy = 1.36
Significance = 10

Residents of Aegean Islands are common customers
Entropy = 2.09
Significance = 40

Young people are good customers
Entropy = 1.53
Significance = 20

Employees of government companies are bad customers
Entropy = 1.45
Significance = 60

Fig. 10. A sample of a decision list.

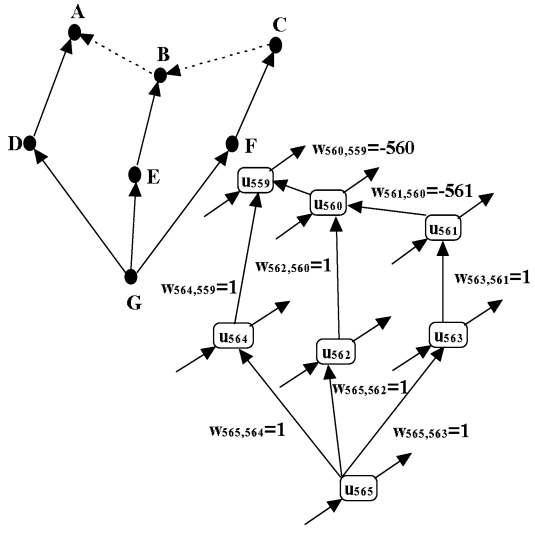

Fig. 11. An example.

- D stands for EMPLOYEES OF GOVERNMENT COMPANIES;
- E stands for RESIDENTS OF AEGEAN ISLANDS;
- F stands for YOUNG PEOPLE;
- G stands from YOUNG PEOPLE, EMPLOYEES OF GOVERNMENT COMPANIES, RESIDENTS OF AEGEAN ISLANDS.

The above inheritance network, in turn, is used to initialize a connectionist network as shown in the same figure. The latter is trained, using relational data as training examples. Finally, refined rules are extracted in the form of a new inheritance network, as shown in Fig. 12. This refined inheritance network can now be used to infer that “Employees

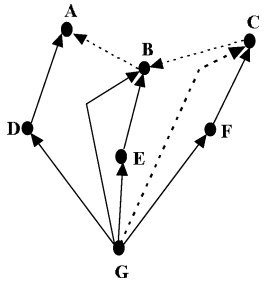

Fig. 12. Resolving conflicts.

of government companies and residents of Aegean Islands are common customers". This holds because of the added positive exception link (G, B) and the negative exception link (G, C). The first supports the fact that Gs are Bs, while the previously conflicted extension (that Gs are not Bs because they are Cs) is removed due to the second.

Notice that every such query, which is actually a combination of attribute values, should be considered as exceptional and refined during training. Hence, combinations of attribute values are considered as potential exceptions, determining a pseudo nonmonotonic domain.

The initial weights of the previous example are shown in Fig. 11. During the refinement phase some of the initial weights are changed while trying to minimize the error function. The error function, given that there are three output cells and one input cell, takes the following form:

$$E = \sum_{i=1}^{3} \left[\sigma(\mathrm{NOUT}_i) - (2\mathrm{DOUT}_i - 1)\right]^2, \tag{17}$$

Thus:

$$\begin{aligned} E = \big[&\big((w_{560,559} \uplus S_{560}) \uplus (w_{564,559} \uplus S_{564}) - (2\mathrm{DOUT}_{559} - 1)\big)^2 \\ &+ \big((w_{561,560} \uplus S_{561}) \uplus (w_{562,560} \uplus S_{562}) - (2\mathrm{DOUT}_{560} - 1)\big)^2 \\ &+ \big((w_{563,561} \uplus S_{561}) - (2\mathrm{DOUT}_{563} - 1)\big)^2\big]. \end{aligned}$$

A table with 30000 records is used as a training set. This table is the same as the one used by the CN2 algorithm when the rules of Fig. 10 are extracted. 31 records of this table are of the form:

"young_people, empl_gov_co, res_Aegean_Isl, bad"

which are encoded as:

$$in_{565} = 1,\ \mathrm{DOUT}_{561} = 0,\ \mathrm{DOUT}_{560} = 0,\ \mathrm{DOUT}_{559} = 1,$$

129 records of them are encoded as:

$$in_{565} = 1,\ \mathrm{DOUT}_{561} = 0,\ \mathrm{DOUT}_{560} = 1,\ \mathrm{DOUT}_{559} = 0,\ \text{and}$$

108 records of them are encoded as:

$$in_{565} = 1,\ \mathrm{DOUT}_{561} = 1,\ \mathrm{DOUT}_{560} = 0,\ \mathrm{DOUT}_{559} = 0.$$

Table 1
Instances of the trace of execution

$w^0_{563,561}$	$w^0_{561,560}$	$w^0_{560,559}$	E
1	−561	−560	1280
1	−561	566	2104
1	−561	−566	1280
566	−561	−560	2144
−1	−561	−560	1072
−1	−566	−560	1072
−1	2	−560	1112

Some epochs of the refinement process are exhibited in Table 1. Theoretically, the weight $w_{560,559}$ is changed properly in the interval $[-\zeta, \zeta]$, where ζ is a "big" positive integer. We consider ζ as the number assigned to the last cell in topological ordering plus one. Actually, weight $w_{560,559}$ is changed taking values from the set $\{-566, \ldots, +566\}$. Since there is not any minimization of the error function, the algorithm proceeds to changing other weights.

Eventually, weight $w_{563,561}$ is changed. A decrease minimizes the error function. Meanwhile, in this case, as possibly in the general case, all outputs are deactivated and there is no conflict resolution at all. In such a case, we ignore the decrement of the error function and we proceed while keeping the change in the weight. This is also the case, when more than one outputs are activated. According to our experience, and in view of all the tests that we have made, the latter case has not been encountered.

Then, weight $w_{561,560}$ is changed. An increase in the weight minimizes the error function with respect to its initial value. More changes do not further minimize the error function, therefore the refinement phase finishes. The revised inheritance network is shown in Fig. 12. There was an increase in weight $w_{561,560}$ that identifies the extension $G \to E \to B$ as prevalent. Therefore a positive exception link (G, B) is added that supports this prevalent extension. Notice that the added link does not affect the existed negative exception link (B, A), which should be retained. Moreover, there was a decrease in weight $w_{563,561}$. Therefore, a negative exception link (G, C) is also added.

Notice that in the revised inheritance network a negative exception link (G, A) is not added, in order to directly exclude the extension $(G \to D \to A)$. Meanwhile, the positive exception link (G, B) can be used to resolve the conflict, indirectly, through adopting a skeptical view [26]. Alternatively, such conflicts can be resolved at the implementation level. Thus, if a positive exception link is added by the extraction algorithm then negative exception links are also added to the rest of the output nodes.

Initial knowledge, represented in the inheritance network shown in Fig. 11, consists of general rules that refer to values of different attributes characterizing the training set (nodes D, E, F), while the input node G represents a potential exception that refers to a combination of attribute values. If we consider all possible general rules, each for a different attribute value, that arbitrarily classify examples to all three classes (A, B and C), we can use ANNNs to refine this initial knowledge, thus resolving these artificial

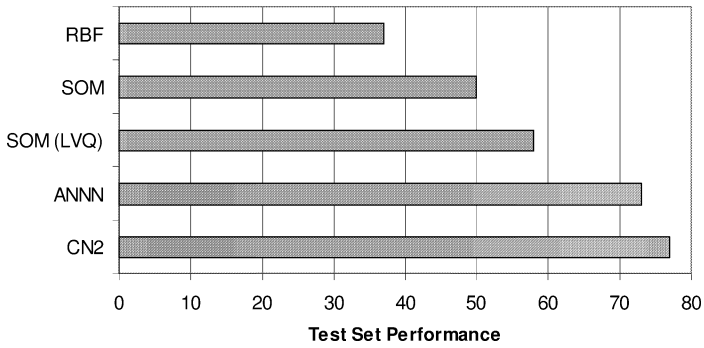

Fig. 13. Test-set performance.

conflicts. Obviously, we can then consider exceptions of these rules, each of them formed by a combination of values, taken from a pair of attributes, and we can try to resolve any conflicts. We can repeatedly consider further exceptions (exceptions of larger *depth*), each of them formed by a combination of values taken from a triple of attributes (as in Fig. 11) and so on. Thus, at the end, the obtained inheritance network represents classification rules, with resolved conflicts, and can be used in a classification process.

We have compared ANNNs, in extraction of customer profiles, to symbolic and connectionist techniques. We have chosen CN2, a well-known symbolic algorithm, and two genuine connectionist learning techniques, namely the Self-Organized Map algorithm [28] and the Radial Basis Function networks. The overall classification accuracy of tested techniques is shown in Fig. 13, using a set of 30000 patterns, chosen out of a set of 60000 patterns. In each case we formed a training set choosing, randomly, 80% of the initial set and a test set from the remaining 20%.

Using SOM, the best results was obtained when the number of neurons was in the range 50 to 150. Within that range the classification performance was not strongly dependent on the number of neurons. The training set size should be large. For instance, for 10000 training patterns, taken from the training set, the obtained performance was 36%. However, for 30000 training patterns and more (up to 50000 patterns that we tried) the classification performance was not varied significantly and it was about 46%. When the class information is appended as input to the training patterns, the classification performance is improved by about 2–4% (it becomes about 48% to 50%). The LVQ phase, with the supervised refinement of the class boundaries, further improves the classification results to 56–58%.

Radial Basis Function (RBF) networks were also used, taking as centers the weight vectors of the neurons obtained after the SOM or LVQ. Then the RBF network training algorithm learns locally the classification function, by accounting mostly the patterns that fall near each training center [20]. However, practically, the numerical solution of the formulated equations for the RBF training becomes problematic, due to the large size of the problem. The currently achieved performance with the RBF networks is only 37%. The poor performance can be attributed to the large size of the training set that involves large matrices, which are mostly sparse.

As far as the symbolic algorithm CN2 is concerned, it overcomes the low classification performance problem of connectionist techniques. Actually, the construction of rules is related to a predefined measure of classification performance (entropy). However, it suffers from a very bad time and space complexity.

The example of the previous section, shown in Fig. 11, concerns a particular combination of attribute values determining a potential exception that must be probably resolved. This example is actually a subset of a complete solution provided by the other mentioned approaches. The overall time complexity is dominated by the number of different combinations of the values of a subset of input attributes to be examined. This is

$$\mathcal{Z} = \sum_{i=2}^{S} \left[\sum_{j=1}^{\binom{S}{i}} \left[|A_{k1}| \times \cdots \times |A_{kj}| \right] \right], \tag{18}$$

where S is the number of attributes and $|A_{kn}|$ is the number of different values that can be assigned to an attribute (a field). Therefore, the time complexity of the particular application of ANNNs to a classification problem is directly proportional to the number of different attributes and to the number of the different values of these attributes. On the other hand, the number of attributes participating in a combination, that is the *depth of exception*, has an impact on the classification accuracy. Large depths tend to overspecialize.

6.2. Analyzing DNA sequence

We, also, evaluate ANNNs using the promoter recognition and the splice-junction determination problem. Initial knowledge of each problem is the rule sets used in [12, 56]. The initial knowledge for the promoter recognition problem is represented by the NIN shown in Fig. 14, where every sequence location is arbitrarily connected both to a "promoter" output node and to a "no promoter" one. These output nodes are connected via an exception link. The same approach is, also, adopted for the splice-junction determination problem.

We used *cross-validation* as the testing methodology. More specifically, for promoter recognition we used leaving-one-out cross-validation in which the set of 106 examples is permuted and divided into 106 sets with only one example in each set. In each evaluation phase, one set, which has never been seen during learning, is used for testing, while

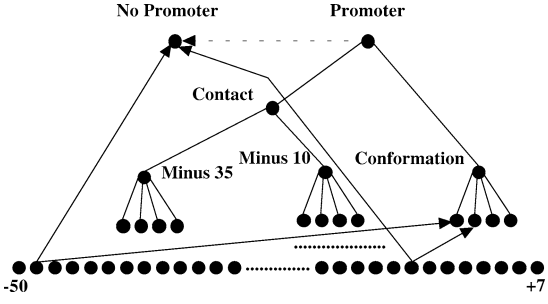

Fig. 14. The initial NIN for promoter recognition problem.

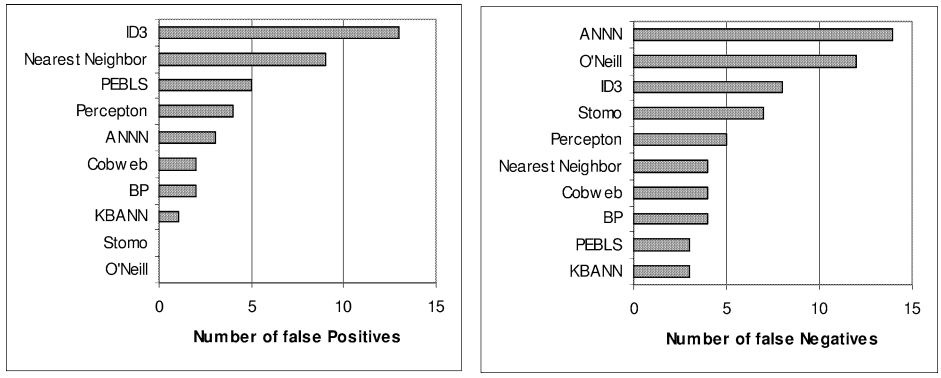

Fig. 15. Test-set performance in promoter recognition.

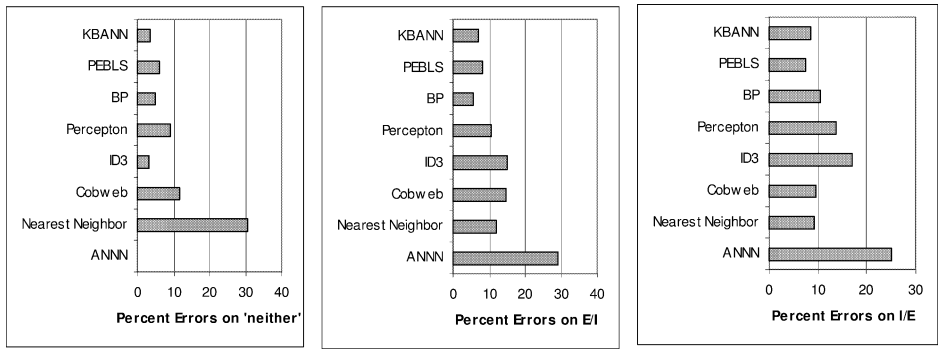

Fig. 16. Test-set performance in splice-junction determination.

the examples of the remaining 105 sets are used for training. Hence, evaluation process requires 106 training phases. Test-set performance in promoter recognition for ANNNs is shown in Fig. 15, where the number of not recognized positive and negative examples is depicted.

Test-set performance in splice-junction determination, using 10-fold cross-validation in 1000 examples, chosen randomly out of the standard set of 3190 examples, is illustrated in Fig. 16. Test-set performance for ANNNs is compared to other empirical learning algorithms and two methods (Stormo, O'Neill [56]) suggested by biologists. Notice that we replicate tests for these algorithms performed in [56].

The overall classification accuracy of ANNNs, in promoter recognition and splice-junction determination, compared to other systems that learn strictly from examples, as well as from examples and background knowledge, is shown in Fig. 17 and Fig. 18, respectively. Notice that we replicate tests for these algorithms performed also in [12].

Performance of ANNNs, in promoter recognition, is tested using, apart from leaving-one-out, both 10-fold and 20-fold cross validation. Fig. 17 shows that the 10-fold methodology exhibits the best results, while the 20-fold the worst. One hypothesis to

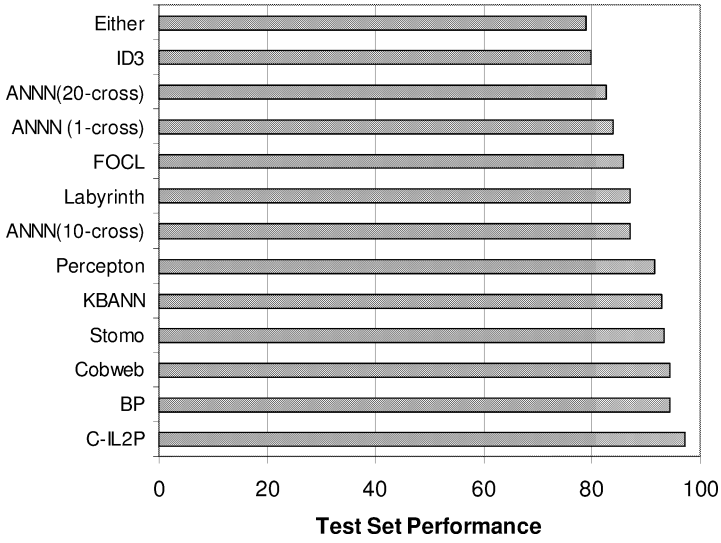

Fig. 17. Classification accuracy in promoter recognition.

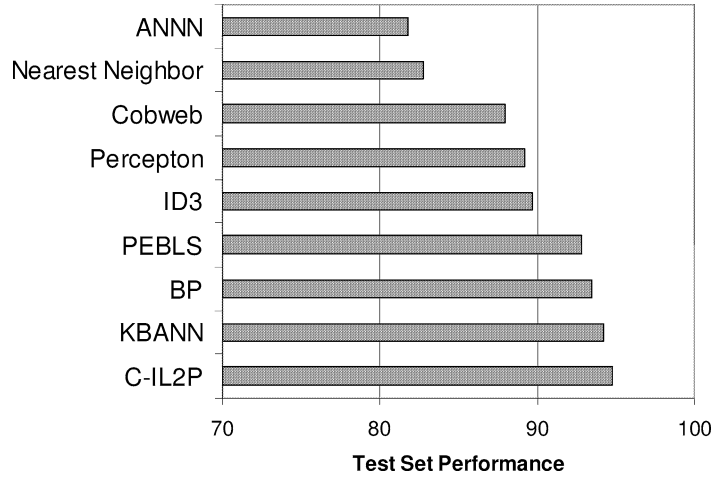

Fig. 18. Classification accuracy in splice-junction determination.

explain this result is that there is a trade off between the decrease of accuracy, due to insufficient training examples (20-fold) or overspecialization (leaving-one-out), and the increase of accuracy, due to an optimum training set (10-fold).

We, also, evaluate the generalization ability of ANNNs in learning from small sets of examples. The testing methodology consists in splitting the initial set of 106 examples of the promoter recognition problem into two subsets, one containing approximately 25% of the examples (26) and the other the remaining examples (80). The latter set is further

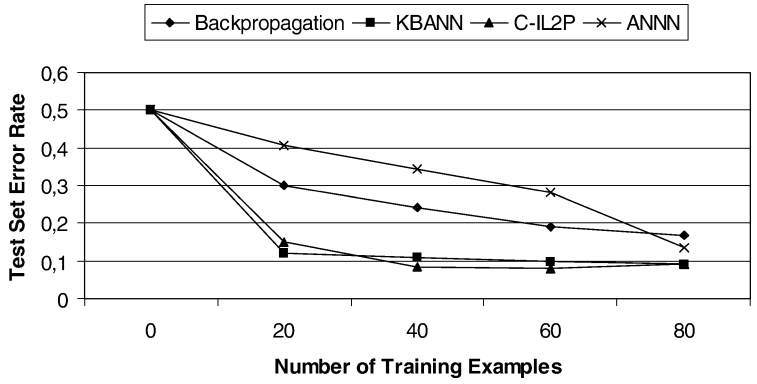

Fig. 19. Classification accuracy in learning from small sets.

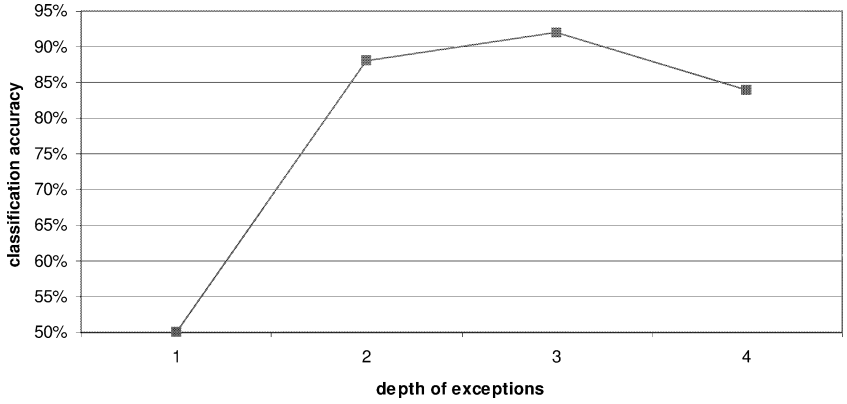

Fig. 20. Classification accuracy as a function of depth of exceptions.

partitioned into sets of increasing size, with the smaller sets being subsets of the larger ones. In each evaluation phase, one of those subsets is used for training, while the set of the 26 examples is always used for testing. The classification accuracy of ANNNs, in each phase, is depicted in Fig. 19 compared to other hybrid systems and backpropagation. Notice that we replicate tests performed in [12]. The same result holds, also, for the splice-junction determination problem.

Depth of exceptions has a straightforward impact on the classification accuracy of ANNNs (in the particular context of the pseudo nonmonotonic domain). On the other hand, depth of exceptions has a straightforward impact on the time complexity. Note that large depths of exceptions are not desirable in such a pseudo nonmonotonic domain, since they introduce overspecialization. On the contrary, they are desirable in a pure nonmonotonic domain, in order to capture exceptions.

Also, notice that all the tests presented so far have been performed with depth equals to 2 (combinations of two attributes). Of course, in a nonmonotonic domain, the larger the depth the better the classification accuracy and the worse the time complexity. In Fig. 20,

classification accuracy, for the promoter recognition problem, is depicted as a function of depth of exceptions, where approximately 25% of the examples (26) were used for testing and the remaining examples for training.

7. Discussion, conclusion and further research

Hybrid systems are typically used for refinement of the domain knowledge. After the refinement phase, the trained Artificial Neural Network, as part of the hybrid system, can be used, as a reasoning system, in order to access the refined knowledge.

In general, an Artificial Neural Network, by construction, supports access operations of the ISA type queries, that is it infers about whether an object possesses a particular property or not. Notice that the complexity of ISA queries in an Artificial Neural Network is comparable to the complexity of ISA queries in path-based networks, when using very effective compression techniques (see, for example, [2]). Of course, this is true if we consider the enumeration of the activation function of unit cost. Moreover, an Artificial Neural Network can support access operations concerning the recognition problem, that is it can find all the objects satisfying a particular set of properties. However, in that case such operations are not effective enough.

The main advantages of hybrid systems come from using the symbolic part to access the refined knowledge. Apart from more effective access operations, the symbolic part has also an explanation capability about the generated outputs. In such a utilization of hybrid systems, the refined knowledge has to be extracted in order to feed back the domain knowledge. Therefore, reliability of the extraction phase is a critical factor to the effectiveness of hybrid systems.

We have tried to use a proper initialization method, a proper training method and a proper extraction algorithm for ANNNs. We proved that they preserve the symbolic meaning of the Initial Inheritance Network, so that we can effectively transform the trained Artificial Neural Network into a comprehensible nonmonotonic inheritance network. Moreover, we evaluated ANNNs by applying them to the classification problem for a pseudo nonmonotonic domain, where exceptions in the initial knowledge are artificially defined. We have followed this evaluating procedure in order to obtain comparative results, because, to our knowledge, there are no available benchmarks for pure nonmonotonic domains. On the other hand, for a monotonic domain there are well known and widely used benchmarks (e.g., see [56]). We empirically proved that, despite ANNNs being capable of refining initial knowledge of a nonmonotonic domain, their performance in the classification problem is comparable to other hybrid monotone systems and better than various well-known monotone pure symbolic or connectionist systems. Of course, none of the monotone systems, that ANNNs were compared to, can be used in classifying examples that belong to a nonmonotonic domain. The main problem, observed during experiments, is that ANNNs, inherently, treat noise as exceptions and thus tend to overspecialize.

We are currently applying ANNNs to different data sets that can constitute a pure nonmonotonic domain (incomplete data, data with drifting or evolving concepts, data arriving over time, etc.). We, also, try to improve the performance of ANNNs in monotonic domains that are reduced to pseudo nonmonotonic domains, as those used in the presented

experimental tests. To this end, we try to identify any malfunctions in cases that biased and noisy data exists, where ANNNs, in pseudo nonmonotonic domains, usually exhibit a lower performance compared to other systems.

In general, ANNNs, treating noise as exceptions, are very sensitive to noise in both the training and the domain knowledge, especially when ANNNs are applied to a monotonic domain. This result is, also, obvious from the presented experiments. We propose to attack this problem by properly selecting the training set, to be able to define as much desired outputs as possible. That is, training is forced to contain not only examples for resolving extensions or deleting exceptions, but also examples that support the already represented knowledge. Moreover, although a continuous analogous refinement process may be more powerful for tackling noise, however it does not satisfy the prerequisites imposed by the proposed approach. To this end, we intent to modify our approach so that continuous analogous refinement processes could be applied instead.

In conclusion, ANNNs are defined to be capable of refining initial knowledge of a nonmonotonic domain. We applied ANNNs to a monotonic domain, reducing it to a pseudo nonmonotonic domain, namely the extraction of classification rules from large relational databases. Notice that, since common monotonic knowledge representation schemes, such as production rules of traditional Expert Systems, are weaker than nonmonotonic inheritance networks, the domain knowledge of a hybrid system can be the domain knowledge of a traditional Expert System. In that case, the initialization-refinement-extraction cycle can be considered as a knowledge acquisition and inferring methodology for traditional Expert Systems, capable of justifying the produced conclusions.

Finally, we are interested in examining the behavior of training methods that do not preserve the symbolic meaning of the initialized Artificial Neural Network, such as those that are not restricted to changing the weights, but can also add hidden layers and connections. The semantics of NINs that are extracted from such trained Artificial Neural Network is under research.

Acknowledgements

We wish to thank the anonymous referees for their constructive comments, suggestions, and invaluable criticisms which helped us to improve the paper. We also wish to thank Dr. S. Papadimitriou and Mr. G. Petalas for their invaluable help during the tests as well as Dr. P. Peppas for useful discussions.

References

[1] R. Andrews, J. Diederich, A.B. Tickle, Survey and critique of techniques for extracting rules from trained artificial neural networks, Knowledge-Based Systems 8 (1995) 373–389.

[2] B. Boutsinas, On managing nonmonotonic transitive relationships, in: Proc. 8th IEEE International Conference on Tools with Artificial Intelligence (ICTAI), Toulouse, France, 1996, pp. 374–382.

[3] B. Boutsinas, Y.C. Stamatiou, G. Pavlides, Massively parallel support for nonmonotonic reasoning, in: J. Geller, H. Kitano, C. Suttner (Eds.), Parallel Processing for Artificial Intelligence, Elsevier Science, Amsterdam, 1997, pp. 41–67.

[4] G. Brewka, Cumulative default logic: In defense of nonmonotonic inference rules, Artificial Intelligence 50 (2) (1991) 183–205.
[5] P. Clark, T. Niblett, The CN2 induction algorithm, Machine Learning 3 (1989) 261–283.
[6] J.P. Delgrande, T. Schaub, W. Ken Jackson, Alternative approaches to default logic, Artificial Intelligence 70 (1994) 167–237.
[7] J.E. Dennis Jr., R.B. Schnabel, Numerical Methods for Unconstrained Optimization and Nonlinear Equations, Prentice-Hall, Englewood Cliffs, NJ, 1983.
[8] D. Dubois, H. Prade, Possibilistic logic, preferential models, nonmonotonicity and related issues, in: Proc. IJCAI-91, Sydney, Australia, 1991, pp. 419–424.
[9] D. Etherington, Formalizing nonmonotonic reasoning systems, Artificial Intelligence 31 (1987) 41–85.
[10] L. Fu, Introduction to knowledge-based neural networks, Knowledge-Based Systems 8 (1995) 299–300.
[11] S.I. Gallant, Connectionist expert systems, Comm. ACM 31 (1988) 152–169.
[12] A.A. Garcez, G. Zaverucha, The connectionist inductive learning and logic programming system, Appl. Intelligence 11 (1999) 59–77.
[13] G.J. Gibson, F.N. Cowan, On the decision regions of multi-layer perceptrons, Proc. IEEE 78 (1990) 1590–1594.
[14] E.M. Gorwin, A.M. Logar, W.J.B. Oldham, An iterative method for training multilayer networks with threshold functions, IEEE Trans. Neural Networks 5 (1994) 507–508.
[15] R. Goodman, Z. Zeng, A learning algorithm for multi-layer perceptrons with hard-limiting threshold units, in: Proc. IEEE Neural Networks for Signal Processing, 1994, pp. 219–228.
[16] H.W. Güsgen, S. Hölldobler, Connectionist inference systems, in: B. Fronhofer, G. Wrightson (Eds.), Parallelization in Inference Systems, Lecture Notes in Artificial Intelligence, Vol. 590, Springer, Berlin, 1992, pp. 82–120.
[17] S.E. Hampson, D.J. Volper, Representing and learning Boolean functions of multivalued features, IEEE Trans. Systems Man Cybernet. 20 (1990) 67–80.
[18] I. Hatzilygeroudis, J. Prentzas, Constructing modular hybrid knowledge bases for expert systems, Internat. J. Artificial Intelligence Tools 10 (1–2) (2001) 87–105.
[19] I. Hatzilygeroudis, H. Reichgelt, Handling inheritance in a system integrating logic in objects, Data Knowledge Engineering 21 (1997) 253–280.
[20] S. Haykin, Neural Networks, 2nd edn., Macmillan Publishing, 1999.
[21] S. Hölldobler, F. Kurfess, CHCL—A connectionist inference system, in: B. Fronhöfer, G. Wrightson (Eds.), Parallelization in Inference Systems, Lecture Notes in Artificial Intelligence, Vol. 590, Springer, Berlin, 1992, pp. 318–342.
[22] S. Hölldobler, Automated inferencing and connectionist models, Post Ph.D. Thesis, Intellektik, Informatic, TH Darmstadt, 1993.
[23] S. Hölldobler, Y. Kalinke, Towards a massively parallel computational model for logic programming, in: Proc. ECAI-94 Workshop on Combining Symbolic and Connectionist Processing, 1994, pp. 68–77.
[24] S. Hölldobler, Y. Kalinke, H. Störr, Approximating the semantics of logic programs by recurrent neural networks, Appl. Intelligence 11 (1999) 45–58.
[25] S. Hölldobler, Challenge problems for the integration of logic and connectionist systems, Technical Report, WV-99-03, AI Institute, Dept. of Computer Science, Dresden University of Technology, 1999.
[26] J. Horty, R. Thomason, D. Touretzky, A skeptical theory of inheritance in nonmonotonic semantic networks, Artificial Intelligence 42 (1990) 311–318.
[27] Y. Kalinke, Using connectionist term representation for first-order deduction—A critical view, in: F. Maire, R. Hayward, J. Diederich (Eds.), Connectionist Systems for Knowledge Representation Deduction, Queensland Univ. of Tech., 1997.
[28] T. Kohonen, Self-Organized Maps, Springer, Berlin, 1997.
[29] R. Maclin, Learning from instruction and experience: Methods for incorporating procedural domain theories into knowledge-based neural networks, Technical Report, UW CS-TR-95-1285, 1995.
[30] R. Maclin, J.W. Shavlik, Refining domain theories expressed as finite-state automata, in: Proc. 8th International Machine Learning Workshop, San Mateo, CA, 1991.
[31] G.D. Magoulas, M.N. Vrahatis, G.S. Androulakis, Effective backpropagation training with variable stepsize, Neural Networks 10 (1997) 69–82.

[32] G.D. Magoulas, M.N. Vrahatis, G.S. Androulakis, Increasing the convergence rate of the error backpropagation algorithm by learning rate adaptation methods, Neural Computation 11 (1999) 1769–1796.
[33] G.D. Magoulas, M.N. Vrahatis, T.N. Grapsa, G.S. Androulakis, A training method for discrete multilayer neural networks, in: S.W. Ellacott, J.C. Mason, I.J. Anderson (Eds.), Mathematics of Neural Networks, Models, Algorithms and Applications, Kluwer Academic, Boston, 1997, pp. 250–254 (Chapter 42).
[34] D. Makinson, General theory of cumulative inference, in: M. Reinfrank (Ed.), Proc. 2nd International Workshop on nonmonotonic Reasoning, Lecture Notes in Artificial Intelligence, Vol. 346, Springer, Berlin, 1989, pp. 1–18.
[35] V.W. Marek, M. Truszczyński, Nonmonotonic Logic, Springer, Berlin, 1993, pp. 141–187.
[36] G.E. Manoussakis, M.N. Vrahatis, G.S. Androulakis, New unconstrained optimization methods based on one-dimensional rootfinding, in: D. Bainov, A. Dishliev (Eds.), Proc. 3rd International Colloquium on Numerical Analysis, Science Culture Technology Publishing, Oxford Graphic Printers, 1995, pp. 127–136.
[37] J. McCarthy, Epistemological challenges for connectionism, Behavioural and Brain Sciences 11 (1988) 44.
[38] W. McCullough, W.H. Pitts, A logical calculus of the ideas imminent in nervous activity, Bull. Math. Biophysics 5 (1943) 115–133.
[39] J.M. Ortega, W.C. Rheinboldt, Iterative Solution of Nonlinear Equations in Several Variables, Academic Press, New York, 1970.
[40] G. Pinkas, The equivalence of connectionist energy minimization and propositional calculus satisfiability, Technical Report, WU CS 90-03, 1990.
[41] G. Pinkas, Energy minimization and the satisfiability of propositional logic, in: D. Touretzky, J. Elman, T. Sejnowski, G. Hinton (Eds.), Proc. of the Connectionist Models School, Morgan Kaufmann, San Mateo, CA, 1990.
[42] G. Pinkas, Symmetric neural networks and propositional logic satisfiability, Neural Computation 3 (1991) 282–291.
[43] G. Pinkas, Expressing first-order logic in symmetric connectionist networks, in: L.N. Kanal, C.B. Suttner (Eds.), Informal Proc. Internat. Workshop on Parallel Processing for AI, Sydney, Australia, 1991, pp. 155–160.
[44] G. Pinkas, Propositional nonmonotonic reasoning and inconsistency in symmetric neural networks, in: Proc. IJCAI-91, Sydney, Australia, 1991, pp. 525–530.
[45] G. Pinkas, Constructing syntactic proofs in symmetric networks, Advances in Neural Information Processing Systems IV (NIPS91) (1992) 217–224.
[46] G. Pinkas, Reasoning, nonmonotonicity and learning in connectionist networks that capture propositional knowledge, Artificial Intelligence 77 (1995) 203–247.
[47] J. Pollack, Recursive distributed representations, Artificial Intelligence 46 (1990) 77–105.
[48] R. Reiter, A logic for default reasoning, Artificial Intelligence 13 (1980) 81–132.
[49] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning internal representations by error propagation, in: D.E. Rumelhart, J.L. McClelland (Eds.), Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1, MIT Press, Cambridge, MA, 1986, pp. 318–363.
[50] E. Sandewall, Nonmonotonic inference rules for multiple inheritance with exceptions, Proc. IEEE 74 (1986) 1345–1353.
[51] B. Selman, H.J. Levesque, The tractability of path-based inheritance, in: Proc. IJCAI-89, Detroit, MI, 1989.
[52] L. Shastri, Default reasoning in semantic networks: A formalization of recognition and inheritance, Artificial Intelligence 39 (1989) 283–355.
[53] D.J. Tom, Training binary node feed forward neural networks by backpropagation of error, Electronics Letters 26 (1990) 1745–1746.
[54] D. Touretzky, The Mathematics of Inheritance Systems, Morgan Kaufmann, Los Altos, CA, 1986.
[55] D. Touretzky, J. Horty, R. Thomason, A clash of intuitions: The current state of nonmonotonic multiple inheritance systems, in: Proc. IJCAI-87, Milan, Italy, 1987, pp. 476–482.
[56] G.G. Towell, J.W. Shavlik, Knowledge based artificial neural networks, Artificial Intelligence 40 (1994) 119–165.
[57] M.N. Vrahatis, Solving systems of nonlinear equations using the nonzero value of the topological degree, ACM Trans. Math. Software 14 (1988) 312–329.
[58] M.N. Vrahatis, CHABIS: A mathematical software package for locating and evaluating roots of systems of nonlinear equations, ACM Trans. Math. Software 14 (1988) 330–336.

[59] M.N. Vrahatis, G.S. Androulakis, J.N. Lambrinos, G.D. Magoulas, A class of gradient unconstrained minimization algorithms with adaptive stepsize, J. Comput. Appl. Math. 114 (2000) 367–386.
[60] M.N. Vrahatis, G.S. Androulakis, G.E. Manoussakis, A new unconstrained optimization method for imprecise problems, in: D. Bainov, A. Dishliev (Eds.), Proc. 3rd International Colloquium on Numerical Analysis, Science Culture Technology Publishing, Oxford Graphic Printers, 995, pp. 185–194.
[61] M.N. Vrahatis, G.S. Androulakis, G.E. Manoussakis, A new unconstrained optimization method for imprecise function and gradient values, J. Math. Anal. Appl.197 (1996) 586–607.
[62] J.D. Watson, N.H. Hopkins, J.W. Roberts, J.A. Steitz, A.M. Weiner, Molecular Biology of the Gene, Vol. 1, Benjamin Cummings, Menlo Park, CA, 1987.
[63] B. Widrow, R. Winter, Neural nets for adaptive filtering and adaptive pattern recognition, IEEE Computer (March 1988) 25–39.
[64] W. Woods, What's in a link?: Foundations for semantic networks, in: R. Brachman, H. Levesque (Eds.), Readings in Knowledge Representation, Morgan Kaufmann, Los Altos, CA, 1985, pp. 217–241.
[65] Z. Zeng, R. Goodman, P. Smyth, Learning finite state machines with self-clustering recurrent networks, Neural Comput. 5 (1993) 976–990.
[66] Z. Zeng, R. Goodman, P. Smyth, Discrete recurrent neural networks for grammatical inference, IEEE Trans. Neural Networks 5 (1994) 320–330.