Data Mining using Nonmonotonic Connectionist Expert Systems

B. BOUTSINAS [1,3], M.N. VRAHATIS [2,3]
[1] Department of Computer Engineering and Informatics,
University of Patras, GR-26500 Patras, GREECE
[2] Department of Mathematics,
University of Patras, GR-26500 Patras, GREECE
[3] University of Patras Artificial Intelligence Research Center (UPAIRC),
University of Patras, GR-26500 Patras, GREECE
URL:
URL:

Abstract: - An application of Nonmonotonic Connectionist Expert Systems (NCESs) in mining classification rules from large relational databases is presented. NCESs are hybrid learning systems that can acquire symbolic knowledge of a nonmonotonic domain, represented using nonmonotonic inheritance networks. This initial knowledge can be refined using connectionist learning techniques and a set of classified examples. Finally, the refined knowledge can be extracted from the connectionist network and fed back in the initial nonmonotonic inheritance network. On the other hand, data mining techniques are used to extract knowledge, in the form of relationships and patterns, from large databases. Data mining techniques, usually, incorporate symbolic learning algorithms that are capable of discovering classification rules from database records that are considered as training examples. In most of the cases, a set of classification rules describes a class through defining a general pattern with exceptions. Once an initial set of such classification rules is known, NCESs can be used in order to refine this set and discover new subtleties.

Keywords and phrases: Data Mining, Neural Networks, Nonmonotonic reasoning.

1 Introduction

Recently, a great attention has been paid in extracting knowledge from large relational databases, either business or scientific ones. Actually, this need is imposed by the explosive growth of such databases and the huge volume of data stored in them. A lot of techniques have been proposed in the literature for such data mining process [6]. There are different kinds of knowledge rules that can be extracted from a database. Usually, classification rules are of the most common ones. 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, a set of classification rules describes a class through defining a general pattern with exceptions. A subset of these rules define 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"). This consideration of general pattern and exceptions is independent from the representation scheme of the classification rule (e.g. "if-then" rules and decision trees).

On the other hand, exceptions introduce nonmonotonicity in a reasoning process. Nonmonotonic reasoning is an important feature of systems that try to mimic common sense reasoning. A lot of different formalisms are proposed, in the literature, that are capable of representing knowledge with exceptions (e.g. [2, 7, 12, 13]).

Nonmonotonic Connectionist Expert Systems, NCESs, [3], are hybrid systems combining connectionist learning techniques with symbolic nonmonotonic knowledge representation schemes. A great attention has been paid, recently, to the development of hybrid systems that use Connectionist Networks as example based learning systems and propositional rules as knowledge representation scheme for the domain knowledge [1, 17]. NCESs use inheritance networks, as a nonmonotonic multiple inheritance knowledge representation scheme for the domain knowledge, and connectionist networks, as a leaning mechanism. The latter is used in order to

refine the initial domain knowledge using connectionist learning techniques and a set of classified examples. The refined knowledge can be extracted from the connectionist network and fed back in a revised nonmonotonic inheritance network.

Therefore, if some initial rules, representing general patterns are known, they can be considered, along with their possible exceptions, as a domain knowledge of an NCES. During a refinement phase the initial knowledge is revised using the database records as training examples. The refined knowledge is extracted and fed back to an inheritance network that reveals the correct exceptions. Since initial general patterns can reflect the experience of an expert, the whole process is an expert-guided data mining technique.

In the rest of the paper we first describe briefly NCESs and then we propose a new data mining methodology. The latter is presenting in conjunction with the knowledge extraction phase of NCESs.

2 Nonmonotonic Connectionist Expert Systems-NCESs

NCESs possess a domain knowledge represented by a nonmonotonic inheritance network. *Nonmonotonic Inheritance Networks* (NINs) is a nonmonotonic multiple inheritance scheme allowing exceptions in the inheritance [16]. Knowledge is represented by attaching to each node of a direct 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. If an exception exists in the inheritance, this is indicated by an exception link.

The domain knowledge of NCESs is used for the initialization of a connectionist network that is used as an example-based learning mechanism in order to refine 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).

The goal of the knowledge initialization phase is to construct a connectionist network that provides the same answers with the nonmonotonic inheritance network that represents the domain knowledge. The constructed connectionist network is trained using a set of classified examples, in order to refine the initial knowledge. The latter is achieved through changing the initial weights. Finally, the refined knowledge is extracted from the connectionist network and is represented by a revised nonmonotonic inheritance network, which replaces the initial one.

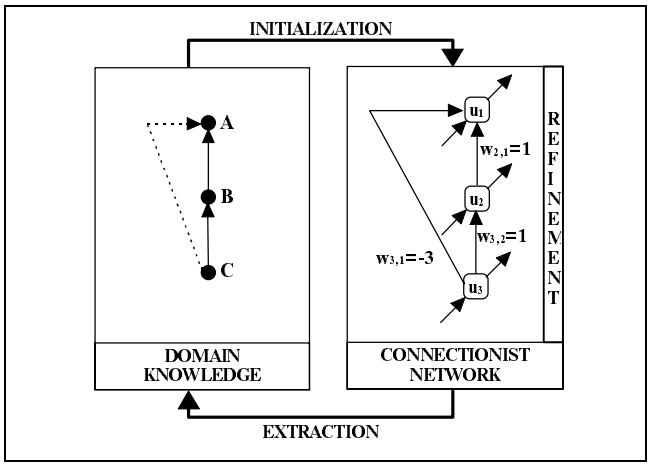

Figure 1: The cycle of knowledge initialization, refinement and extraction

2.1 The knowledge initialization phase

The goal of the knowledge initialization phase is formally defined as:

Given: a direct acyclic graph $G(V, E = R \bigcup POS \bigcup NEG)$ with V the set of nodes that represent objects of the domain of discourse and E the set of edges that represent relations between those objects, where the set R consist of the edges that represent ordinary relations, the set POS consists of the edges that represent exceptional positive relations, the set NEG consists of the edges that represent exceptional negative relations and $R \bigcap POS \bigcap NEG = \emptyset$.

Initialize a connectionist network N, with: (cell inputs and activations are discrete)

- a set of cells $U = \{u_1, \ldots, u_k\}$, each one of which receives a network binary input ($[0, 1]$) and gives a network binary output ($[-1, 1]$). Moreover, each cell u_i performs an operation S, to be described later, on its inputs.
- a set of integer weights $W = \{w_{i,j} | i, j \leq k\}$,
- a proper activation function σ applied to the network outputs.

The initialization of a connectionist network utilizing symbolic knowledge with exceptions, requires the simulation of the canceling of inheritance represented by the negative links (denoted by a dot line).

Exception links that represent exceptions in the inheritance of properties, take precedence over the normal links that represent inheritance from the more general classes to more specific ones. There is also an ordering of the exception links according to how specific the class they refer to is. The more specific is the referring class the higher priority has the exception link.

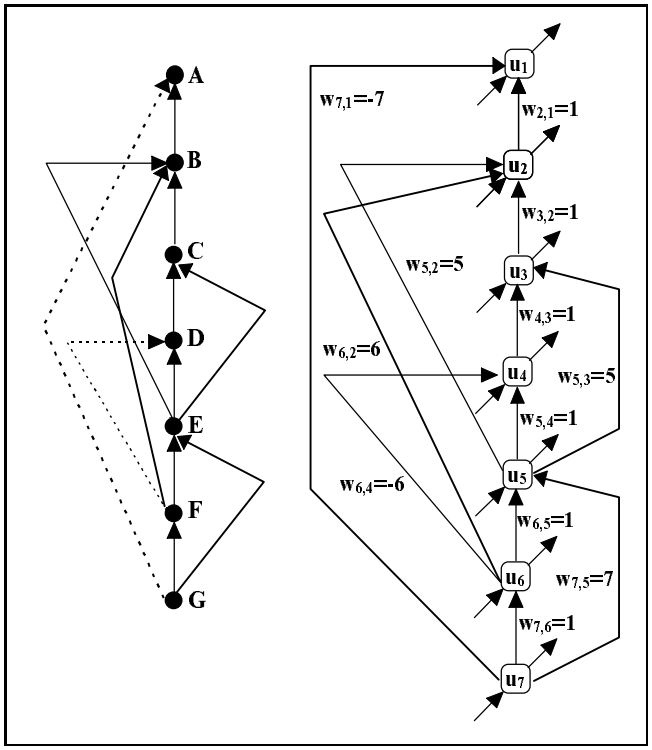

Figure 2: Assigning negative or positive weights according to a topological ordering

To properly initialize the connectionist network we have to preserve this "inferential distance ordering" [16]. To this end, we defined a cell operation S as the computation of the maximum absolute value of the inputs of the cell. Thus,

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

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

$$x \uplus y = \frac{x - \varepsilon + y}{|x - \varepsilon + y|} \times \frac{||x| - |y|| + |x| + |y|}{2},$$

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

$$\mathrm{in_j} = \begin{cases} 0, & \text{if} \quad S_j = 0; \\ w_{i,j} \uplus S_j, & \text{if} \quad S_j \neq 0. \end{cases}$$

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 where there are more than one inputs of equal absolute value, the output is identical to the negative input. This is achieved by subtracting a very small positive number ε from someone of the inputs determining the resolution of distinguishing x and y.

More specifically, we assign to a negative or a positive link a negative or a positive weight, 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 to each cell an integer, 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 $f(u_i) < f(u_j)$. The above methodology [3] is applied to the example of Fig. 2.

2.2 The knowledge refinement method

The knowledge refinement is based on the training of the corresponding connectionist network using a set of training examples. 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 connectionist network. In our case, this is accomplished by restricting the changing of the weights to selected ones and by using a fixed architecture.

This method can be applied to selected weights in order to minimize the corresponding error function E without utilizing the gradient of E. In our case the error function is defined by

$$E = \sum_{i=1}^{C} \Big[\sigma(\mathrm{NOUT}_i) - (2\mathrm{DOUT}_i - 1)\Big]^2,$$

where σ is a discrete activation function [8], C refers to the number of cells that corresponds to the semantically related input cells, NOUT_i is the output of the ith cell and DOUT_i is its corresponding desired output. Notice that C is much less than the total number of cells of the network.

The proposed training method is applied by changing selected weights at each epoch, which is very useful in our approach. It is based on a recently proposed optimization method [18] and it can be applied to imprecise problems since the only computational information required by it is the algebraic sign of E.

3 General patterns and their exceptions in classification rules

A set of classification rules is actually a discretization of a continuous domain into classes. Most often, such a discretization is established in different levels of generalization. More generalized

levels concern discretization into general classes, while less generalized levels concern discretization into classes that are specifications of the general classes. We can consider these specifications as being introduced by exceptions to general patterns (notice the example in the introduction). This notion of general patterns and their exceptions can be considered in all of the most known representation schemes for classification rules.

Firstly, consider Decision Lists [11] as a propositional-like representation scheme that is used in some of the most effective data mining algorithms (e.g. [4]). Decision Lists representation scheme is actually an ordered set of "if-then" rules. The antecedents of the rules is a conjunction of conditions and the consequent is a class. Thus, every rule represents a description of a class. Actually, a class description consists of a set of such rules. This set describes a class through defining a general pattern (a rule) with exceptions (additional rules). Exceptions correspond to the early rules in the Decision List, while general patterns correspond to the later ones. Thus, the notion of general patterns and their exceptions is obvious in decision lists.

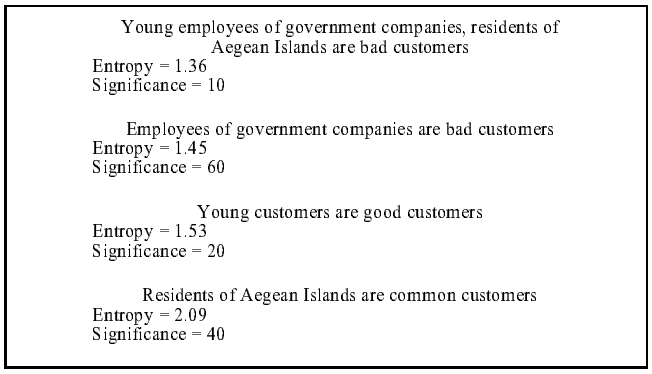

Figure 3: A sample of a Decision List

A sample of rules of a Decision List is shown in Fig. 3. These rules are constructed by the CN2 algorithm 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. The last three rules define general patterns while the first one is an exception to the last two rules.

Data mining algorithms construct classification rules along with a measure of *interestingness* for each rule. This measure is evaluated through a quality function that guides the search process and guarantees that the most interesting rules will be chosen from the search space. For instance, CN2 data mining algorithm [4] uses a pair of measures in order to represent the quality of a rule: *entropy*, an information theoretic measure, and *significance*, a statistical measure. The first refers to the accuracy of the rule in predicting a class, with respect to the training set (the lower the entropy, the better the accuracy). The latter guarantees that this accuracy is not obtained due to randomness (higher significance corresponds to less possibility that the accuracy is not obtained due to randomness). The entropy of the last two rules is high enough; so we can conclude that there exist a lot of exceptions to the general patterns. Actually, there exist more exceptions to the last rule. On the other hand, the significance of all the rules guarantees that the rules are not obtained due to randomness.

Another representation scheme is the *Inductive Logic Program* (ILP). ILP is a *First Order Logic* (FOL)-like representation scheme with more expressive power than propositional-like representations. Definitions of the classes are represented by formulae of FOL. Although FOL formulae does directly support representation of exceptions, we can identify the notion of general patterns and their exceptions in ILP. Actually, there exist certain ILP systems that support such a nonmonotonic setting (e.g. [10]). In these cases, general patterns can be identified in formulae-rules with more than one class in the conclusion, while formulae-rules with only one class in the conclusion can be identified as exceptions to general patterns.

A sample of rules generated by the ILP system CLAUDIEN [10] is shown in the following:

- b1a(X) $\bigvee$ b1b(X) $\leftarrow$ perlodidae(X,A)
- b1a(X) $\leftarrow$ perlodidae(X,A),rhyacophilidae(X,B)

The rules concern the biological classification of river water quality through monitoring the existence of special organisms, the benthic macroinvertebrates. The predicates b1a(X) and b1b(X) represent classes of water quality of the sample X and the predicates of the form family(X,A) represent the existence of family in sample X at the abundance level A. The whole set of rules consists of 79 rules and only 28 involve the presence of a single conclusion like the last rule [5]. General patterns that can be identified are "Perlodidae family at abundance level A denotes water quality class B1A" and "Perlodidae family at abundance level A denotes water quality class B1B", while there is an exception to the latter general pattern that is:

[1]The classes in the rules are shown changed, with respect to their actual values, for safety reasons

"Perlodidae family at abundance level A along with Rhyacofilidae family at abundance level B denote water quality class B1A".

Finally, we consider the *Decision Trees* [9]. A tree graph structure is used in order to represent class definitions. Every node of the tree represents an attribute while the edges emanating for this node represent the possible values for the attribute. Nodes that are leaves represent the different classes. Therefore, a complete path of this tree, a path beginning from the root to a leave, consist of nodes that represent values to certain attributes which, in turn, form a definition of the class represented by the leaf of this path.

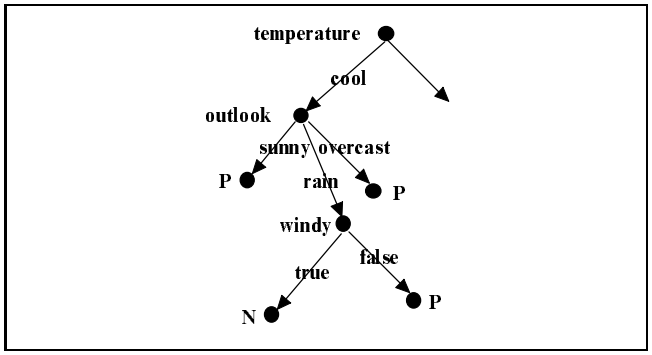

Figure 4: A Decision Tree

We can identify the notion of general patterns and their exceptions in decision tree. Meanwhile, this is not as obvious as in the case of decision lists. To see this, let us first define the concept of a broom. A broom is a set of complete paths resulting from a path $\{ROOT, N_1, N_2, \ldots, N_i\}$ along with all the paths in the subtree rooted at N_i. If most of paths of a broom represent the same class C, we can identify as a general pattern the definition of C formed by the nodes $\{ROOT, N_1, N_2, \ldots, N_{i-1}\}$. This general pattern can be considered along with exceptions formed by the paths of the broom that do not represent class C.

Consider, for example, the decision tree shown in Fig. 4. The tree concerns the classification of "Saturday Mornings" into suitable and not suitable ones for a certain unspecified activity [9], according to some environmental attributes. The path *temperature*, *outlook* along with all the paths in the subtree rooted at *outlook* define a broom. Most of the paths in this broom represent class P. Therefore, we can identify the general pattern "Cool Saturday Mornings are suitable". Meanwhile there is a path that represents class N. Therefore, we can identify the exception "Cool, rainy and windy Saturday Mornings are not suitable".

4 The proposed methodology

We can conclude, from the previous section, that the notion of general patterns and their exceptions is inherent in data mining process. 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.

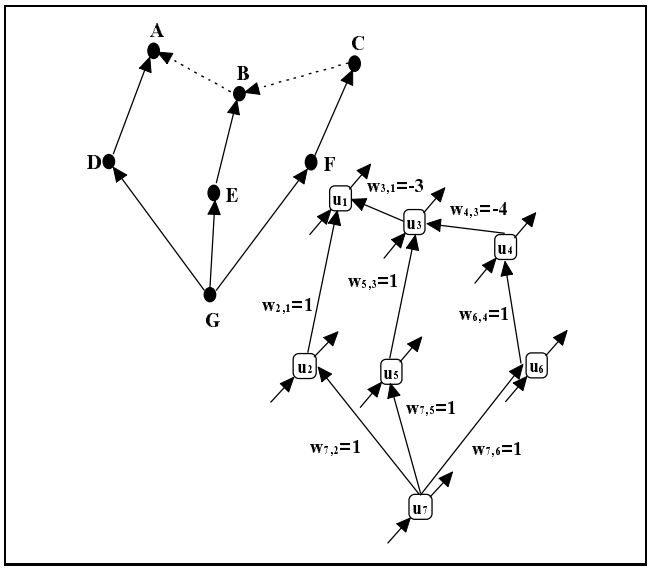

Figure 5: An example

Rules representing general patterns can be defined by experts and they are used, along with their possible exceptions, to form an initial inheritance network. The latter is used to initialize a connectionist network that is trained using relational data. Finally, refined rules are extracted in the form of a new inheritance network.

Assume, for example, that an expert defines the three general patterns mentioned in the previous section. An initial inheritance network is constructed as shown in Fig. 5, taking under consideration the possible exception G. The interpretation of the symbols is as follows: A stands for *BAD*, B stands for *COMMON*, C stands for *GOOD*, D stands for *EMPLOYEES OF GOVERNMENT COMPANIES*, E stands for *RESIDENTS OF AEGEAN ISLANDS*, F stands for *YOUNG CUSTOMERS* and finally G stands from *YOUNG EMPLOYEES OF GOVERNMENT COMPANIES RESIDENTS OF AEGEAN ISLANDS*. The previous 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. 6. This refined inheritance network can be used in answering whether "Employees of government companies, residents of Aegean Islands are bad customers".

The proposed method that extracts the refined rules heavily relies on reversing the initialization phase. The cells of the connectionist network are, simply, transformed into nodes of the NIN. But the refinement of the initialized knowledge is actually represented by the changes in the weights of the connections of the connectionist network. These changes impose the refinement of the initialized knowledge, resolving conflicts. The initial knowledge of the inheritance network is changed due to insertions of exception links.

4.1 Resolving conflicts

Nonmonotonic multiple inheritance frequently introduces multiple extensions of a theory. 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 [15], and there is no effort to resolve the conflicts, as it is the case in [14].

Using NCESs, during the knowledge refinement phase, the conflicts can be resolved in favor of the most probable ones, with respect to the training set. In resolving conflicts, a technique is used based on the identification of the extension that is supported by the set of classified examples used during the refinement phase. 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 example of Fig. 5 the initial weights are shown there. During the refinement phase some of the initial weights are changed trying to minimize the error function:

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

since there are three output cells of concern. Thus:

$$\begin{aligned} E = \Big[& \Big((w_{2,1} \uplus S_2) \uplus (w_{3,1} \uplus S_3) - (2\mathrm{DOUT}_1 - 1) \Big)^2 \\ & + \Big((w_{5,3} \uplus S_5) \uplus (w_{4,3} \uplus S_4) - (2\mathrm{DOUT}_3 - 1) \Big)^2 \\ & + \Big((w_{6,4} \uplus S_6) - (2\mathrm{DOUT}_4 - 1) \Big)^2 \Big]. \end{aligned}$$

A table with 100000 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. 3 are extracted. 7271 records of this table are of the form:

<"young employees of government companies, residents of Aegean Islands","bad">

that is transformed as:

$in_7 = 1, DOUT_4 = 0, DOUT_3 = 0, DOUT_1 = 1,$

2437 records of them transformed to the form:

$in_7 = 1, DOUT_4 = 0, DOUT_3 = 1, DOUT_1 = 0,$

1293 records of them transformed to the form:

$in_7 = 1, DOUT_4 = 1, DOUT_3 = 0, DOUT_1 = 0.$

Epochs of the refinement process are exhibited in the following table (for details see [3]):

$w^0_{6,4}$	$w^0_{4,3}$	$w^0_{3,1}$	E	h
1	−4	−3	77664	2
3	−4	−3	77664	−2
−1	−4	−3	44004	
−1	−4	−3	44004	5
−1	1	−3	44004	−5
−1	−9	−3	44004	10
−1	6	−3	54348	
−1	−4	−3	44004	4
−1	−4	1	44004	−4
−1	−4	−7	44004	8
−1	−4	5	29840	

In the initialized connectionist network prevalent extensions are those represented by 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 of a weight, since it is toward canceling a negative weight, gives precedence to a prevalent extension supporting a positive result. An increase or a decrease of 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 this inheritance network, a proper exception link, positive or negative depending on the supported by the extension result. The added exception link concerns the extension as a whole and, clearly, not the existed, in the initial inheritance network, exception link whose the corresponding connection has changed. Therefore, the added exception link is attached to the path that represents the extension.

Consider, for example, the trained connectionist network of Fig. 6. There is an increase in the weight $w_{3,1}$ that identifies the extension $G \rightarrow D \rightarrow A$ as prevalent. Therefore a positive exception link (G, A) is added that supports this prevalent extension. Notice that the added link does not affect the existed negative exception link (B, A), that has to remain. Moreover, there is a decrease in the weight $w_{6,4}$. Therefore, a negative exception link (G, C) is also added.

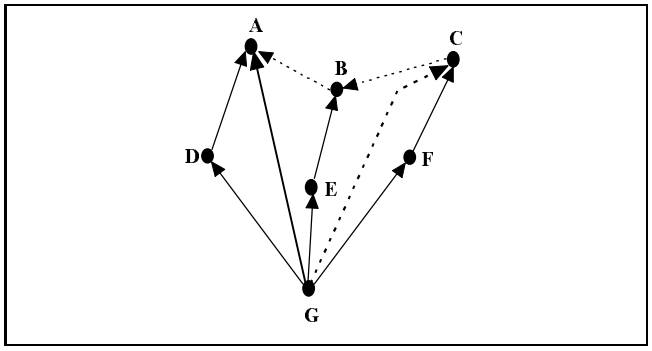

Figure 6: Resolving conflicts

In order to add properly exception links 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 existed exception link, we consider that an extension is represented by a path containing the tail and head nodes of this exception link. Such a path is the $G \rightarrow E \rightarrow B \rightarrow A$ in Fig. 6, containing the tail node B and the head node A of the existing exception link (B, A).

In general, if there exists a path from the node representing the input cell under consideration to the node representing the output node under consideration, 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.

4.2 The extraction algorithm

To attack both of 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, in the case of resolving conflicts, if there are extensions represented by paths without exceptions, then, since only the weights that correspond to exception links are considered, the identification of the prevalent extension is not possible. Actually, in such a case the initialized connectionist network can not be trained.

We cope with this problem by allowing the addition of weights to the set of selected weights that correspond to ordinary links. These are the weight of connections that are adjacent to cells representing nodes without incoming exception links. Notice that adding exception links, in the case of resolving conflicts, is related to the implementation of the inheritance network.

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 shown in the following:

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

Revise the Inheritance Network N, 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:

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

5 Conclusion

A new methodology for mining classification rules from large relational databases is presented. The main advantages of the proposed methodology are

a) it exhibits a high accuracy in the classification process,

b) it is simple and it always gives the answer with certainty,

c) it gives the ability to the user to directly manipulate the mining process.

We are currently applying the proposed methodology in real life applications and our experience is that it is a robust and effective one.

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, vol.8, No.6, Elsevier Science B. V., 1995, pp.373-389.

[2] B. Boutsinas, Y. C. Stamatiou, G. Pavlides, "Massively Parallel Support for Nonmonotonic Reasoning", "Parallel Processing for Artificial Intelligence", J. Geller, H. Kitano, C. Suttner, Eds., Elsevier Science Publishers B.V., North-Holland, 1997, pp.41–67.

[3] B. Boutsinas, M.N. Vrahatis, "Nonmonotonic connectionist expert systems", Recent Advances in Circuits and Systems, N.E. Mastorakis Ed., World Scientific Publishing Co. Pte. Ltd., 1998, pp.404–412.

[4] P. Clark, T. Niblett, "The CN2 Induction Algorithm", Machine Learning, vol.3, No.4, 1989, pp.261-283.

[5] S. Dzeroski, "Inductive Logic Programming and Knowledge Discovery in Databases", "Advances in Knowledge Discovery and Data Mining", U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, Eds., 1996, pp.117–152.

[6] U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, Eds., "Advances in Knowledge Discovery and Data Mining", MIT Press, 1996.

[7] J. Horty, R. Thomason, D. Touretzky, "A Skeptical Theory of Inheritance in Nonmonotonic Semantic Networks", Artif. Intell. vol.42, pp.311-318, 1990.

[8] G.D. Magoulas, M.N. Vrahatis, T.N. Grapsa and G.S. Androulakis, "A training method for discrete multilayer neural networks", Mathematics of Neural Networks, Models, Algorithms and Applications, S.W. Ellacott, J.C. Mason and I.J. Anderson Eds., Kluwer Academic Publishers, Boston, Chapter 42, 1997, pp.250–254.

[9] J.R. Quinlan, "Induction of Decision Trees", Machine Learning, vol.1, 1986, pp.81-106.

[10] L. De Raedt, M. Bruynooghe, "A Theory of Clausal Discovery", Proceedings of the IJCAI-93, San Mateo, Calif., 1983, pp.1058-1063.

[11] R. Rivest, "Learning Decision Lists", Machine Learning, vol.2, No.3, 1987, pp.229-246.

[12] R. Reiter, "A logic for default reasoning", Artif. Intell. vol.13, 1980, pp.81-132.

[13] E. Sandewall, "Nonmonotonic Inference Rules for Multiple Inheritance with Exceptions", Proceedings IEEE 74, 1986, pp.1345–1353.

[14] L. Shastri, "Default Reasoning in Semantic Networks: A Formalization of Recognition and Inheritance", Artif. Intell. vol.39, 1989, pp.283-355.

[15] D. Touretzky, "The Mathematics of Inheritance Systems", (Morgan Kaufmann, Los Altos, CA), 1986.

[16] D. Touretzky, J. Horty, R. Thomason, "A Clash of Intuitions: The Current State of Nonmonotonic Multiple Inheritance Systems", Proceedings of the IJCAI-87, Milan, Italy, 1987, pp.476-482.

[17] G.G. Towell, J.W. Shavlik, "Knowledge Based artificial neural networks", Artif. Intell. vol. 40, 1994, pp.119-165.

[18] M.N. Vrahatis, G.S. Androulakis, G. Manoussakis, "A new unconstrained optimization method for imprecise function and gradient values", J. Math. Anal. Appl., vol.197, No.2, 1996, pp.586–607.