RULE_OUT METHOD: A NEW APPROACH FOR KNOWLEDGE EXPLICITATION FROM TRAINED ANN

Löic Decloedt [1], Fernando Osório [1,2], Bernard Amy [1]

[1] Laboratoire LEIBNIZ / LIFIA - IMAG -INPG
46, avenue Félix Viallet 38031 Grenoble Cedex 1 - FRANCE
E-mail: osorio@imag.fr - amy@imag.fr
Web: http://leibniz.imag.fr/RESEAUX/

[2] UNISINOS - Computer Science Dept.
Av. Unisinos, 950 - CP 275 - CEP 93022-000 RS- BRAZIL
Web: http://www.unisinos.tche.br/

ABSTRACT

Artificial Neural Networks have been applied in many different domains with success. Their main advantage come from their generalisation ability on a learned base of examples. On the other hand, their major drawback comes from their lack of justification abilities. So, several methods have been designed to explicit their knowledge in a symbolic form, in order to get it comprehensible for a human operator. We present here an ANN knowledge explicitation system, included in a more general neuro-symbolic hybrid AI system: the INSS system. Usually, existing explicitation systems attempt to express the whole part of the ANN knowledge. Here, choice has been made to represent only its more significant part. In order to do this, we have developed methods based on the incrementality of the applied ANN learning method, and on network simplification. We present these methods in this paper; then, some experimental results obtained on the Monk's problems are shown.

1. INTRODUCTION

Artificial Neural Networks (ANN) have shown very good abilities to represent knowledge expressed into a set of examples, and interesting generalisation properties. However, their sub-symbolic representation, expressed in terms of units structure containing weighted links, doesn't allow to explain their answers. Several methods, defined as ANN knowledge extraction methods, have been developed to express the ANN representation in terms of symbolic knowledge (i.e. symbolic rules). As we don't consider that ANN knowlegde is embedded in the network, but expressed in a way that is difficultly comprehensible for a human, we'd rather talk of knowledge explicitation than knowledge extraction.

Unlike usual explicitation systems, that work on the whole ANN knowledge, we started from the idea to only express the **more significant** part of considered knowledge. The aim of this concept is, on one hand, to enhance comprehension of a human user, and on the other hand, to reduce the complexity of such process.

We implemented this explicitation method as a part of an hybrid AI system, that combines both symbolic and connectionist models: the INSS system [11]. This system is presented in the first part of this paper, in Section 2. The main principles of the explicitation method are exposed in the next section. Then, methods for selecting more significant parts of considered knowledge are presented in Section 4. At least, we present some experimental results obtained on a well known machine learning problem: the Monk Problem [13].

2. INSS SYSTEM

The acquired knowledge of a given problem can be represented in different forms. It can be expressed either as theoretical concepts, or as practical examples. To take advantage of the whole knowlegde, it would be interesting to be able to use together both aspects. The INSS System - Incremental Neuro-Symbolic System - developed by F. Osorio [11], is an hybrid system that combines theoretical and practical models of knowledge (i.e. symbolic rules and pratical examples). Its base model (inspired by KBANN system [14] with an important improvement in the ANN learning algorithm) works with two independent modules, each one representing one of the considered aspects.

Theoretical aspect is represented by a set of production rules that associate hypothesis conditions or facts to conclusion facts. This set of rules is manipulated by a Symbolic Module. Practical aspect is modelled by an Connectionist Module, composed of an ANN. In INSS system, ANN learning is incremental, i.e. it changes the network topology, working both on modifying output weights, and adding new hidden units in order to represent the new acquire knowledge. The algorithm used to perform this task is based on Cascade Correlation, developed by S.E. Fahlmann [4]. This kind of algorithm (i.e. incremental learning methods) improves significantly the learning ability of KBANN networks, e.g. see [9].

In addition to this, communication processes between these modules, Symbolic one and Connectionist one, are designed to reduce the gap that can exist between the two knowledge aspects. These processes work on information exchange, based on translation from one representation to the other one. Rules compilation in ANN form represents communication from Symbolic Module to Connectionist Module; knowledge explicitation allows to express an ANN in a theoretical model, and thus represents communication form Connectionist to Symbolic Module. This particular aspect of INSS, knowledge explicitation, is described in more details in the next sections.

3. MAIN ASPECTS OF KNOWLEDGE EXPLICITATION

At the moment, two main types of methods exist in the domain of knowledge explicitation [1]. On one hand, there exits "decompositional" methods that approximate each unit of the ANN as a propositional variable. Then, they attempt to find logical relationships between these variables, through weights value carried by links between corresponding units. At least, rules are derived from these relationships. This type of method is represented by algorithms like SUBSET algorithm, created by L.M. Fu [5], or NofM algorithm [14]. On the other hand, "pedagogical" methods don't examine each unit of the net, rather considering it as a "black box". These methods try to directly find the relationships between the inputs and the outputs of the network, through directed search into the space of inputs. Examples of this kind of method are "Extraction-as-learning", developed by M. Craven [2], or that developed in BRAINNE system, by S. Sestito and T. Dillon [12].

However the major drawback of all these methods is that they focuses on the whole knowledge contained by networks. This aspect increases the processing complexity in a significant manner, especially for decompositional algorithms. Furthermore, the corresponding number of obtained rules can alter their abilities to be understood by a human operator.

The fact of we have chosen to represent more significant parts of considered knowledge, instead of the whole part of it, allows us to reduce both complexity of such process and number of expressed rules. This is obtained by selecting the more significant elements of the network, and applying the explicitation only on these elements. As explicitation process is only applied on particular elements on the considered network, decompositional methods have been chosen to realise it. Indeed, this kind of method can be applied to particular parts of the network, whereas pedagogical methods usually works on the whole net. Chosen methods are SUBSET and NofM algorithms. In the next section, we show how more significant parts of treated network are selected, prior to explicitation.

4. SELECTING NETWORK ELEMENTS

In this part, we present methods that are used to select elements representing more significant network knowledge. Firstly, it is shown how explicitation can be focused more particularly on knowledge that the network has acquired during learning. Secondly, it is seen how relative importance of units on global outputs can be measured. At least, some methods to evaluate relative importance of weights are presented.

4.1. Selecting acquired-knowledge-linked elements

As seen in §2, INSS combines two models of knowledge, a symbolic one and a connectionist one. Studies has shown that learning is improved if initial theoretical knowledge is inserted into ANN before [6][10][14]. In the INSS system, theoretical knowledge insertion can be realised by rules-to-network compilation. Such network, obtained by compiling a set of rule, can then be submitted to the learning of a set of examples. After learning, due to Cascade Correlation algorithm [4], ANN possesses two kinds of separeted knowledge: initial knowlegde and knowledge acquired during learning. Then, it would be interesting to be able to express only the second type of knowledge, and not the whole one. It would then allow us to focus on this type of knowledge, and to compare it to the initial knowledge. It can be noted that this possibility is absent from every ANN extraction system we have seen up to now.

Since the initial theoretical knowledge is conserved during learning, the INSS system learning algorithm is incremental, i.e. it works by adding new hidden units. Indeed, initial hidden weights are frozen before learning, and cannot be changed. Thus, knowledge acquired during learning is represented by new units (output and hidden units), added by Cascade Correlation algorithm. Then, explicitation process can be applied directly to this type of unit, in order to express additive knowledge acquired during learning.

4.2. Selecting significant units

As, during explicitation process, each unit is expressed as a propositional variable, it can be useful to detect significant units before applying it. Indeed, if only the most significant variables are involved, resulting rules can be simplified, and comprehensibility increased. Methods for measuring unit significance have been developed. The aim is then to remove from the network least significant units. One method measures influence of network inputs. We have created another method that allows to detect units with few decisive outputs.

4.2.1. Influence of inputs

This method is designed to measure how global error is affected, when a particular network input value is set to be always equal to its mean value over a set of examples. It is inspired by a method developed by John Moody [8], that has developed a measure, called *sensitivity*, which measures this effect. This measure, realised on a set of N patterns is defined, for each input i, as:

$$Si = \frac{1}{N}\sum_{j=1}^{N}\frac{\partial SE(j)}{\partial x_i(j)} \quad .dx_i(j)$$

where $x_i(j)$ represents value of input i for pattern j, SE(j) represents global Squared Error for pattern j, and $dx_i(j)$ is defined as:

$$d\,x_i(j) = \left(x_i(j) - \bar{x}_i\right)$$

where $\bar{x}_i$ is the mean value of input i over the set of examples.

4.2.2. Detecting few decisive units

We have created another simple method to detect units which have nearly the same output values in the network input space [3]. When detected, this kind of unit can be approximated as always sending the same output ; it then can be removed from the network, after having propagated its mean activation through the network, by modifying bias of units which have an incoming link from this unit.

To detect this type of unit, the activation frequency (i.e. the frequency with which a given unit sends an output value near 1) of each unit is measured over a set of examples. If the considered set of examples is representative enough, then a unit with high activation frequency will be approximated as a unit always sending value of 1, and a unit with very low activation frequency will be approximated as a unit sending output of 0.

4.3. Selecting significant weights

Some methods have been designed to measure influence of weights on network outputs, in order to prune those having the least influence. This aspect is very important, because the complexity of explicitation methods that are used is strongly related to the number of incoming weights of units they are applied to. The number of expressed rules can be reduced in a significant manner too. Two different methods have been designed to measure this influence.

4.3.1. Relative percentage

One assumption is made that, considering connections incoming to the same unit, links that have *very low* weights with respect to other ones has little influence on the output of the considered unit. This hypothesis can be contested, but our experimental results has shown that Cascade Correlation algorithm assigned very low weights to less important links. So, we have created a very simple method that measure relative importance of each weight with respect to other ones incoming to the same unit [3]. In order to do that, we calculate percentage of each weight with respect to global sum of the others weights. Then, weights having the least percentage can be pruned in priority. In our system we provide some feedback to the user so he can check whether this operation does affect global error.

4.3.2. Influence on global error

The assumption that the influence of a link is proportional to its weight can be wrong: if very little weights usually have little influence, it has been observed, after learning, links with big weights that were not significant for global network response. So, another method, inspired by Optimal Brain Damage algorithm [7], has been developed to measure influence of each weight on global error; it is based on calculus of global error second derivative. This measure, called *saliency*, is defined, for a weight w_k as:

$$S(w_k)=\frac{1}{2}\frac{\partial^2 ASE}{\partial w_k \partial w_k}.(w_k)^2$$

where ASE represents the training Average Squared Error. Then weights with low saliency can be pruned, and explicitation can be greatly improved by this way.

5. EXPERIMENTAL RESULTS

Experiments have been realised, using an artificial machine learning problem known as the Monk's Problem. Its main interest is that it had been tested out on many learning methods [13], and is therefore useful to compare INSS system to other ones. This problem is defined as following; we consider robots having the following characteristics:

```
HEAD_SHAPE: ROUND, SQUARE, OCTAGON;
BODY_SHAPE: ROUND, SQUARE, OCTAGON;
IS_HOLDING: FLAG, SWORD, BALLOON;
JACKET_COLOUR: RED, BLUE, YELLOW, GREEN;
HAS_TIE: YES, NO;
IS_SMILING: YES, NO;
```

In the first Monk's Problem data set, valid robots are those having the same head shape as their body shape, or a red jacket. Corresponding rules can be seen below.

```
Monk1<- HEAD_SHAPE=ROUND, BODY_SHAPE=ROUND              (I)
Monk1<- HEAD_SHAPE=SQUARE, BODY_SHAPE=SQUARE            (II)
Monk1<- HEAD_SHAPE=OCTAGON, BODY_SHAPE=OCTAGON          (III)
Monk1<- JACKET_COLOUR=RED                               (IV)
```

Each rule having the form *(Y<- X1, X2,, ..., Xn;)* means: "*if (X1 and X2 and ... Xn) then Y*". Each set of rules having the form *(Y<-X1; Y<- X2; ... Y<-Xn;)* means: "*if (X1 or X2 or ... Xn) then Y*".

This set of rules can then be compiled into an ANN. The corresponding ANN has one output and, for each input variable, one input for each possible value, i.e. 17 inputs in total. Each input is coded as: <Variable#Value>. For example, the input representing (HAS_TIE = YES) is coded as (HAS_TIE#YES).

Chosen test has been realised by selecting 50% of initial theory, i.e. rules I and II, and by compiling them in the form of an ANN. Afterward, this network has been submitted to the learning of a set of examples containing the whole knowledge (i.e. the original Monk's 1 problem *learning* data set). Knowledge explicitation processes have then been activated to verify whether missing knowledge had been acquired during learning. Resulting net after learning is represented in Figure 1.

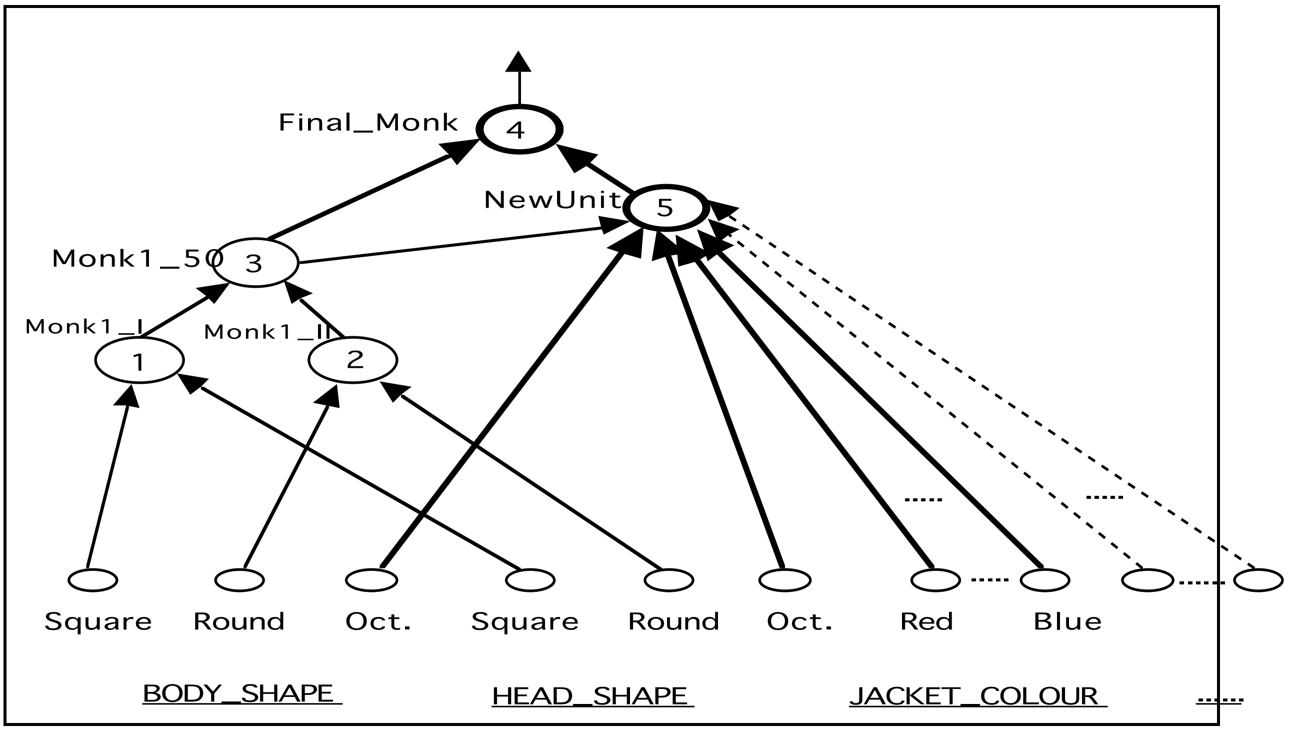

Figure 1 - Resulting ANN structure after learning

Units 1 to 3 represent initial knowledge. The incoming weights of these units has remained unchanged during learning. Unit 4 represent the network global response, and had been added during the first step of the learning algorithm execution, unit 5 is the unit added by Cascade Correlation algorithm. Newly acquired knowledge during learning phase is represented by units 4 and 5. So explicitation will focus on these particular units, as seen in §4.1.

Incoming links to these units represented with a dashed line had a very low weight (less than 1% of the total weight sum), and have been eliminated as discussed in §4.3.1. The link represented with a simple line had a big weight, but its saliency were very low (cf. 4.3.2). then, it has been pruned too. So, only incoming links with "strong" lines were remaining before applying explicitation process. The explicitation has been performed with the SUBSET algorithm [5][14], the resulting rules are:

```
Final_Monk<- Monk1_50%;
Final_Monk<- not (NewUnit);
NewUnit<- not (JACKET_COLOUR#RED), not (BODY_SHAPE#OCTAGON);
NewUnit<- not (JACKET_COLOUR#RED), not (HEAD_SHAPE#OCTAGON);
```

Monk_50% variable is equivalent to initial rules (I) and (II). We define a new variable Neg_NewUnit such as: Neg_NewUnit= not (NewUnit). Then we introduce it in the above rules, using De Morgan formulas:

```
Final_Monk<- Monk1_50%;
Final_Monk<- Neg_NewUnit;
Neg_NewUnit<- JACKET_COLOUR#RED;
Neg_NewUnit<- BODY_SHAPE#OCTAGON, HEAD_SHAPE#OCTAGON;
```

It can be seen that missing knowledge was found again. It can be noted that use of incrementality and simplification processes increased very significantly the efficiency of the explicitation process. Indeed, execution time of explicitation, using SUBSET algorithm, without simplification is approximately equal to two or three hours on a Sun Sparc station, whereas it takes few seconds if simplification is realised before. Simplification also allows to reduce number of resulting rules by a factor of ten to twenty, without loss of generality. This is very useful to check main aspects of explicited knowledge. At least, use of incrementality allow us to separate initial knowledge and acquired knowledge during learning; as a result, comprehensibility is enhanced.

6. CONCLUSION

We presented in this paper an ANN knowledge explicitation system based on incrementality and simplification. This system is able to do incremental rule extraction because, taking advantage of incremental learning characteristics, it allows to focus extraction only on newly acquired knowledge. It can be noted that this aspect is absent from ANN explicitation systems that we have seen up to now. As more significant parts of considered ANN are selected, prior to explicitation, this system is simplification-based; few systems provide this possibility. Experimental results showed that, given an initial network that had been compiled from an incomplete set of rules and then submitted to learning, missing knowledge could be found back very efficiently using this system.

However, it has been observed that it was more difficult to extract significant features from network containing no initial knowledge. Indeed, this knowledge was expressed in a very dense way, so it was difficult to detect more significant terms. A solution to this problem could be to work out an incremental symbolic knowledge acquisition method. This method would consist in, starting from an empty, or almost empty, initial set of rules, extending it through several compile-learn-extract loops. Each loop would consist in compiling current set of rules and submitting resulting net to learning. Then we could extract simpler rules from this ANN to insert them into initial set of rules. This process could then be applied back to the new set of rules until entire knowledge is acquired. We are currently developing studies in this direction.

Another problem comes from the fact that considered knowledge has strong symbolic requirements that remain implicit for the network. For example, in case showed in §5, as inputs of the net are nominal variables, input units representing the same variable are mutually exclusive, and, at any time, one of these inputs has a valid value. As these properties are implicit, explicitation system doesn't take them in account, and give, for example, rules containing conjunction of different values for the same variable in antecedent. These rules are valid at the network level, but have no reality in the symbolic context in which we are. For examples, rules described on §5 were not obtained directly under the given form, but have been submitted before to some symbolic manipulations, giving an equivalent set of rules, taking in account these symbolic properties. So, we would need to use symbolic methods to automate these manipulations, in order to enhance comprehensibility.

Works that will be developed in the future, for this explicitation system, concern compile-learn-extract loops research, designing methods for the manipulation, validation and verification of extracted rules, and use of this system on real-world problems.

BIBLIOGRAPHY

[1] Andrews, R.; Diederich, J.; Tickle, A.B. *A Survey and Critique of Techniques for Extracting Rules form Artficial Neural Networks* ; To appear: Knowledge-Base Systems. Queensland University of Technology. 1995.

[2] Craven, M. W. *Using Sampling and Queries to Extract Rules from Trained Neural Network;* Machine Learning: Proc. of the 11th International Conf. of San Francisco. 1994.

[3] Decloedt, L. *Explicitation de Connaissances dans un Système Hybride d'Intelligence Artificielle.* Computer Science DEA Research Report, LIFIA - IMAG, Grenoble - France, 1995.

[4] Fahlman, S. E.; Lebiere, C. *The Cascade-Correlation Learning Architecture*; Carnegie Mellon University, Technical Report - CMU-CS-90-100. 1990.

[5] Fu, L. M. *Integration of Neural Heuristics into Knowledge-Based inference* ; Connection Science 1(3): 325-339. 1989.

[6] Giacometti, A. *Modèles hybrides de l'expertise* ; Ph.D. Thesis, LIFIA - IMAG, Grenoble - France, 1992.

[7] LeCun, Y.; Denker, J. S.; Solla, S.A. *Optimal Brain Damage* ; Advances in Neural Information Processing Systems 2, D.S. Touretzky ed., Morgan Kauffmann publishers, 1990.

[8] Moody, J. *Prediction Risk and Architecture Selection for Neural Networks* ; From Statistics to Neural Networks: Theory and Pattern Recognition Applications, NATO ASI Series F. Springer-Verlag, 1994.

[9] Opitz, D. W.; Shavlik, J. W. Dynamically Adding Symbolically Meaningful Nodes to Knowledge-Based Neural Networks. To appear: Knowledge-Base Systems. Computer Science Dept. Univ. of Wisconsin-Madison. 1995.

[10] Orsier, B. *Etude et application de systèmes hybrides neurosymboliques* ; Ph.D. Thesis, UJF - LIFIA, Grenoble - France, 1995.

[11] Osorio, F. *INSS: A Hybrid Symboli-Connectionist System that Learns from Rules and Examples* ; (in portuguese), Panel'95 - XXI Latin American Conference on Computer Science, Canela, Brazil, 1995.

[12] Sestito, S.; Dillon, T. *The Use of Sub-Symbolic Methods for the Automation of Knowledge Acquisition for Expert Systems and Their Applications* ; Proc. of the 11th International Conference on Expert Systems and their Applications (AVIGNON'91), Avignon. 1991.

[13] Thrun, S. B. et al.*The Monk's Problem - A Performance Comparison of Different Learning Algorithm* ; Carnegie Mellon University, Technical Report CMU-CS-91-197. 1991.

[14] Towell, G. *Symbolic Knowledge and Neural Networks: Insertion, Refinement and Extraction* ; Ph.D Thesis, University of Wisconsin-Madison - Comp. Science Dept. 1991.