

A new neuro-symbolic hybrid system for rule extraction or rule refinement

Jihane Boulahia-Smirani¹ and Laurent Bougrain²

¹ ENSI/LIA, Artificial Intelligence Group, Tunis, Tunisia
l.smirani@ttnet.tn

² LORIA/INRIA-Lorraine, Villiers-lès-Nancy, France
bougrain@loria.fr

Abstract. We present a new hybrid neuro-symbolic system allowing rule extraction and rule refinement. This approach combining various modules including a connexionist module, a symbolic module, a rule insertion module and a rule extraction module. The system can start with or without *a priori* knowledge expressed as rules. There are two main benefits from introducing rules in the network : improving generalization to new instances and simplifying learning. It presents an important improvement in comparison with just a connectionist system because of its knowledge discovery capability. The model provides a new approach applicable to machine learning with high-performance.

1 Introduction

The concept of hybrid systems is very broad [1]. This group of applications includes any method, which merge two different approaches to solve a problem. It's often suggested that traditionally serial symbol processing systems of artificial intelligence (AI) and inherently massively parallel artificial neural networks (ANNs) offer two radically, perhaps even irreconcilably different paradigms for modeling minds and brains, both artificial as well as natural. We can note that the two paradigms have forces and weaknesses, which are often complementary. Rather than taking an entirely new-route, it is a question of re-using the assets, from where needed to integrate the two paradigms, thus the birth of the hybrid systems. The basic assumption in a neuro-symbolic hybrid system is that the symbolic system models and connectionist ones are not sufficient by them-selves [2], but their combination can allow the execution of all types of cognitive operations. Such a justification is a very general one. We would like to have a more precise justification about the real contribution of the hybrid approach. What exactly provides the combination of neural networks and Knowledge-based systems? Researches claim that hybrid systems take advantage of their respective component strengths. In this paper, we are interested in the neuro-symbolic hybrid system. Our contribution is system named HLS : Hybrid Learning System combining a connexionist module with a symbolic module.

2 HLS : Hybrid Learning System

The schematic diagram of the overall architecture (Fig.1) has been proposed for learning task. It consists of four major components : a connectionist module, a symbolic module, a rule extraction module, and a rule insertion module. The HLS system provides facilities to transfer rules from a symbolic module to a connectionist one, and extract rules from the connectionist module to the symbolic one [5, 9]. First, *a priori* knowledge defined by the experts of the domain as a rule database is gathered into the rule insertion module. Then, the neural network is initialized inserting the rules. This approach solves two big problems related to artificial neural networks: on one hand this simplifies the choice of the number of units, on the other hand we obtain a good assignment of initial values to the connection weights. The result is a ready to work three layered neural network. After this first stage, the connexionist module is activated to train the neural network. This training is based on a set of examples. After learning the rule extraction module is activated to extract rules from the neural network. The result of this operation is a set of rules relating inputs to outputs. Finally, the rules extracted are transmitted to the symbolic module for validation.

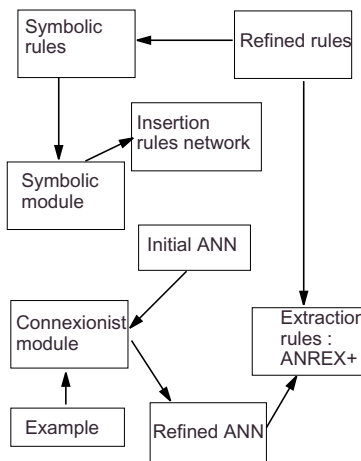


Fig. 1. Hybrid Learning System

2.1 Connectionist module

Which kind of neural neural the connectionist module should contains? The first question that emerges at the time of the conception of such a network is to know the number of the necessary hidden layers. Theoretically, the perceptron (two-layer network) can classify the space of data only linearly. The exactness of this classification can reach 100% if data is linearly separable. However, a multi-layer network can achieve a non-linear classification. In addition, it was shown [13] that any function, which is approximated by a multi-layer perceptron, can also

be approximated by a multi-layer perceptron with only one hidden layer, if its architecture includes sufficient number of hidden units. perceptron. It is known that such a three-layer perceptron can realize arbitrary binary mappings [17]. For the experiments presented in this article, we used a "feedforward" neural network that is trained using the backpropagation algorithm. Hidden units have a sigmoid activation.

2.2 Rule extraction module

It's becoming increasingly known that, without some forms of explanation capability, the full potential of trained artificial neural networks may not be realized [1]. The merits of including rule extraction techniques, as an adjunct to conventional artificial neural network, include the provision of a user explanation capability. In fact, experience has shown that an explanation capability is considered to be one of the most important functions provided by symbolic artificial intelligence systems. In contrast to symbolic AI systems, neural networks have no explicit knowledge representation. A new module has been defined to solve this problem, this module is named ANREX+: A heuristic Algorithm for Neural Rule extraction+. ANREX+ is an improvement of ANREX. The algorithm heuristic for neural Rule Extraction ANREX [9] is going to permit a simple rule extraction from a neural network. The result provided by ANREX is a concise and precise description of the internal working network. A standard three layer feedforward network is the base of the algorithm. Weight decay is implemented while backpropagation is carried out. After the network is pruned, the activation values of the hidden units are discretized. Rules are extracted by examining the discretized activation values of the hidden units.

Clustering Algorithm : Rules are not readily extractable because the activation values of the hidden unit are continuous. The discretisation of these values paves the way for rule extraction. Many clustering algorithms can be used for this purpose. The following algorithm discretises the activation values of a hidden unit.

For each hidden unit

1. Let E be the larger gap allowed from a referenced activation value a (e.g. $E = 0.2$)
2. Start with the activation value a_0 generated by the first example in the training set.
3. Cluster the activation values $| a_1 - a_0 | < E$.
4. Represent this cluster's activation value by the average of the activation values in the cluster
5. Select next a_0 , repeat 3 and 4 for clustering until all activation values are clustered.

When the clustering is done, the network's accuracy is checked. A sufficiently small E guarantees that the network with discretized activation accuracy does

not drop and there are still many discrete values. The clustering can be performed again with a larger E to minimize the number of clusters. Otherwise E should be reduced to a smaller value.

The ANREX Algorithm : Let f be the sigmoid transfer function applied to the weighted sum of connections, val the activation value function, $R1$ the first set of rules, $R2$ the second set of rules and R the final set of rules.

```

/* Rule Extraction between the input and hidden units */
For every hidden unit j do
  For every input unit i (where connection j-i exist) do
    If f(i) = val(j) then
      ith input participate in the rule R1
      Add i with value to R1
    End if
  End For
End For

/* Rule Extraction between the hidden and output units */
For every hidden unit j do
  For every output k (where connection j-k exist) do
    If f(j) = k then
      kth output participate to the rule R2
      Add k with value to R2
    End if
  End For
End For
Merge (R1,R2) * R

```

The algorithm for neural Rule Extraction ANREX [8,9] is going to permit a simple rule extraction from a neural network. The result provided by ANREX is a concise and precise description of the internal working network. ANREX has been applied to a multi-layer using the back-propagation algorithm. First ANREX has been applied to character recognition, second we applied it to a speech recognition task and third in Iris database. In the second application, results show that the performance of ANREX decreases. The number of extracted rules become very large and we have observed the existence of many weights with minimal values so we have decide to study methods of pruning in the neural networks. This extension of ANREX is named ANREX+.

The ANREX+ algorithm : ANREX+ is an improvement of ANREX. Before applied ANREX, we pruned the network. 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 the least significant units. 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. Som another method, inspired by Optimal Brain Damage Algorithm [16], 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 : $S(w_k) = (1/2)(\sigma^2 ASE)/\sigma w_k \sigma w_k (w_k)^2$ where ASE represents the training Average Squared Error. Then weights with low saliency can be pruned, and explanation can be greatly improved by this way.

2.3 Insertion rules module

There are two main benefits from introducing rules in the network : improving generalization to new instances and simplifying learning. Towell and shavlik [12,13] have proposed methods for mapping a set of prepositional rules into a neural network, refered to as Knowledge Based Artificial Neural Networks (KBANN). This module is inspired of this method. The result of the translation is a network composed of a set of units linked by weighted connections. Conjunctive rules are translated into a neural network by setting weights on all links corresponding to positive (i.e. unnegated) antecedents to the weight value w , by setting weights on all links corresponding to negated antecedents to $-w$, and the bias on the unit corresponding to the rule's consequent to $(-P+0.5)* w$; where P is the number of positive antecedents to the rule. Experiments shown that the best value of w is $w=4$. To translate a set of rules encoding a disjunction, the insertion module sets the weight of each link corresponding to a disjunctiveantecedent to w and the bias on the unit encoding the consequent to $-w/2$.

Finally, the symbolic module finds out the probably incorrect rules and examples. Imp- rovement of this module is in study in a master project.

3 Experimental results

This section details results we obtained on two classification tasks : first, on a real-world application of speech recognition and second, on the classical benchmark of Iris plant species.

3.1 Speech recognition

The recognition vocabulary consists of 20 Arabic word. Speech was collected from 50 speakers (28 females and 22 males). The speakers were arbitrary separated into a set of 30 speakers (15 females and 15 males) for training and a set of 20 different speakers (13 females and 7 males) for testing [18]. The speech signal is digitizes from a head - mounted microphone. A custom interface decimates the 48 kHz digitized speech signal to 16 kHz. The speech is segmented into hamming - window and processed into feature vectors. Each feature vector consists of 12 cepstral coefficients.

For speech recognition, there are two types of experiments: first we began with testing the time convergence of the system, second the hybrid system recognition with incomplete data. The convergence time of the system is compared with a MLP (Multilayer Perceptron) convergence. The average of the epochs number necessary for the convergence shows clearly that with the HLS system the convergence is more rapid (60 epochs) than the MLP architecture. (980 epochs). The second experiment has to do with the recognition rate of the system with incomplete data as shown in Table 1. This experiment aims at verifying if the system is able to recognize data with incomplete set of examples. In the first test, we created a network with 85% of the examples. Then, we applied the rule extraction module and the insertion module. In a second step, we used another incomplete example set constructed by removing 25% of the examples contained into the complete set. Third a set composed of 65% of the examples, finally only 50% of examples. The results obtained show that HLS is able to deal with this problem (incomplete data). We showed that generalization with HLS is better than generalization obtained with only a neural network.

Table 1. Speech recognition rate

Portion of examples	Recognition with a MLP	Recognition with HLS
100%	100%	100%
85%	82,1%	89,4%
75%	65%	81%
65%	61%	74%
50%	59%	62%

3.2 Iris database

The iris database is a well known database in pattern recognition³. The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant (versicolor, setosa and virginica). One class is linearly separable from the other two. Each example is described by four continuous numerical attributes (X1, X2, X3 and X4 : sepal length, sepal width, petal length and petal width given in centimetres).

We experimented the capabilities of the system with and without inserting a set of rules to initialize the neural network as we describe in section ??.

Rules defined to initialize neural network :

```

Rule 1 : If petal_length < 2.5 then Iris_setosa.
Rule 2 : If 2.5 < petal_length and petal_length < 5
         and 0.75 < petal_width and petal_width < 1.6 then Iris_versicolor.
Rule 3 : If 5 < petal_length and 1.6 < petal_width then Iris_virginica.

```

³ <http://ftp.ics.uci.edu/pub/machine-learning-databases/iris/>

At the beginning of the procedure, 50 fully connected neural networks are trained. Then the networks were pruned until its performance on the training data dropped below 95%. The weights and topology of networks with the smallest number of connections and a recognition rate of greater than 96% were saved for rule extraction. One of the pruned networks has only 2 hidden units and 8 connections with 98.1% accuracy in the training set and 97.2% on the testing set. We applied the clustering algorithm described in section ???. We found only 2 discrete values for each hidden unit. For the first hidden unit, 37 of 50 training examples have activation values equal to 0 and 13 have activation values equal to 1. For the second hidden unit, the activation value of 15 examples is 1 and the activation value of the second hidden unit is equal to -0.52. Since we have two activation values for each hidden unit, four different outcomes at the output unit are possible. So, an example will be classified as Iris setosa as long as its activation value at the second hidden unit is equal to 1. Otherwise, the example is classified as Iris versicolor provided that its first hidden unit activation value $H1=0$. The default class will then be Iris virginica. In this example, only two inputs, X3 and X4, determine the activation values of the second hidden unit, H2. But, since X4 is 1 for all the training data, H2 is effectively determined by X3. Since the weights of the arcs connecting input units X3 and X4 to the second hidden unit are -5.4 and 4.8 respectively, it is easy to conclude that if $X3=0$ then H2 equals 1 otherwise, H2 is -0.52. This implies that an example will be classified as Iris setosa only if X3 is 0 (hence H2 is 1). The activation value of the first hidden unit, H1, depends only on X1 and X2. The weights of the arcs connecting input units X1 and X2 to the first hidden unit are 5.1 and 8.1, respectively hence H1 is 0 if and only if $I26=I34=0$. Other input combinations will yield value 1 for H1. Hence an example with $X1=X2=0$ will be classified as Iris versicolor.

HLS rules are :

```
Rule 1 : If petal_length <= 1.9 then Iris_setosa.
Rule 2 : If petal_length <= 4.9 and petal_width <= 1.6 then Iris_versicolor.
Default Rule : Iris_virginica.
```

We can conclude that rules extracted from neural network with HLS are more precise than the initial rules. In fact, the limit of petal_length in the initial set of rules is 2.5 versus 1.9 in the extracted rules and the limit of petal_length in the initial set of rules is 5 versus 4.9 in the extracted rules. For reference, the set of rules generated by the decision tree C4.5[?] is as follows :

```
Rule 1 : If petal_width <= 0.6 then Iris_setosa
Rule 2 : If petal_length <= 4.9 and petal_width > 0.6 and petal width <= 1.7
        then Iris_versicolor.
Rule 3 : If petal_width > 1.7 then Iris_virginica.
Rule 4 : If petal_length > 4.9 then Iris_virginica.
Default rule : Iris_setosa.
```

Experiments show that rules extracted by HLS and C4.5 are quite comparable. So HLS provides understandable and useful rules. Our system can start with or without priori knowledge. In the second case, our system starts with only examples. The result obtained in this case is similar to result obtained with examples and rules. The difference between the two systems is in the speed of extraction

and generalisation of the neural network. The first configuration is faster but the extracted rules are the same. The difference between the two systems is the subject of research actually.

4 Conclusion

We have proposed and implemented a neuro-symbolic model. The proposed architecture named HLS (Hybrid Learning System) provides a good performance and allows acquisition/extraction of network knowledge. The system can start with or without priori knowledge expressed as rules. The results are very interesting when comparing them to those of a neural network. With HLS extraction rules, neural networks should no longer be regarded as black boxes and with HLS insertion rules in the neural network we improve generalization to new instances and simplify learning even in presence of incomplete data.

References

1. Andrews, R., Diederich, J., Tickle, A.B.: A Survey And Critique of Techniques For Extracting Rules From Trained ANN. Technical Report - Neurocomputing Reasearch Centre, Queensland University of Technology. Brisbane, Australia. January (1995)
2. Hilario, M: An overview of Strategies for Neurosymbolic Integration. In Connectionist Symbolic Integration. From Unified to Hybrid Approaches. Ron Sun (ED.)-Chapter 2. Kluwr Academic Publishers (1996)
3. J.Sima. "Neural expert systems". Neural networks, Vol 8 N:2, pp 261 (1994)
4. Orsier, B.: Etude et application de systèmes hybrides neuro-symboliques. Thèse de doctorat , UJF-LIFIA, Grenoble- France.
5. Giles, C.L., Omlin, C.W.: Extraction, insertion, and refinement of symbolic rules in dynamically driven recurrent neural networks. Connection Science, vol. 5, n3, pp. 307-337 (1993)
6. Angluin, D., Smith, C.H.: A survey of inductive inference : Theory and methods. ACM Comput. Survey, vol, 15, pp. 237-269, Sept. (1983)
7. Rumelhart, D.E., Hinton, G.E., Williams, R. J.: Learning internal representations by error propagation. In parallel distributed Processing : Explorations in the Microstructures of Cognition, volume 1. Cambridge, MA : MIT Press (1986)
8. Mouria Beji, F., Boulahia Smirani, J.: Extraction and Insertion rules during the training process of a neural network. ACIDCA'2000. 22-24 March (2000) Monastir-Tunisia.
9. Mouria Beji, F., Boulahia Smirani, J., Marrakchi, M.: ANREX : an algorithm for neural network rule extraction. In IEEE International Multiconference on Computational Engineering in Systems Applications IEEE-SMC. April (1998)
10. Towell, G., Shavlik, J.W.: Extracting refined rules from knowledge-based neural networks. Machine learning, vol. 13, pp. 71-101 (1993)
11. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann. San Mateo, CA (1993)
12. Shavlik, J. W.: Combining symbolic and neural learning. Machine Learning, vol. 14, n3, pp. 321-331 (1994)

13. Hecht-Nielsen, R.: Theory of the Backpropagation Neural Network. International Joint Conference on Neural Networks, Vol 1, pp 593–605, Washington (1989)
14. Kerber, R.: Chimerge: Discretization of numeric attributes. In AAAI-92, Proceedings Ninth National conference on Artificial Intelligence, pp 123-128. AAAI Press/The MIT Press (1992).
15. Sun, R., Alexandre, F.: Connectionist-symbolic Integration : From Unified to Hybrid Approaches. Lawrence Erlbaum Associates (1997)
16. Le Cun, Y., Denker, J.S., Solla S. A.: Optimal Brain Damage. Advances in Neural Information Processing Systems II (Denver 1989), ed. D.S. Touretzky, pp 598-605, Morgan Kaufmann, San Mateo (1990)
17. Lallement, Y.: Intégration Neuro-Symbolique et Intelligence Artificielle: Applications et Implantation Parallèle. Thèse de Doctorat en Informatique, CRIN-INRIA Lorraine, Université Henri Poincaré - Nancy I, France, Juin (1996)
18. Wu, Z.: Introduction to Experimental Speech. The high Education Press, Beijing. (1987)