To be published in the Proceedings of the 14th International FLAIRS Conference, Key West, Florida, May 2001.

An Efficient Hybrid Rule Based Inference Engine with Explanation Capability

Ioannis Hatzilygeroudis, Jim Prentzas*

University of Patras, School of Engineering
Dept of Computer Engin. & Informatics, 26500 Patras, Hellas (Greece)
&
Computer Technology Institute, P.O. Box 1122, 26110 Patras, Hellas (Greece)
E-mails: ihatz@cti.gr, ihatz@ceid.upatras.gr, prentzas@ceid.upatras.gr
Phone: +30-61-960321, Fax: +30-61-9960322

Abstract

An inference engine for a hybrid representation scheme based on neurules is presented. Neurules are a kind of hybrid rules that combine a symbolic (production rules) and a connectionist representation (adaline unit). The inference engine uses a connectionist technique, which is based on the 'firing potential', a measurement of the firing tendency of a neurule, and symbolic pattern matching. It is proved to be more efficient and natural than pure connectionist inference engines. Explanation of 'how' type can be provided in the form of if-then symbolic rules.

1. Introduction

There have been efforts at combining expert systems and neural networks (connectionism) into hybrid systems, in order to exploit their benefits (Medsker 1994). In some of them, called embedded systems, a neural network is used in the inference engine of an expert system. For example, in NEULA (Tirri 1995) a neural network selects the next rule to fire. Also, LAM (Medsker 1994) uses two neural networks as partial problem solvers. However, the inference process in those systems, although gains efficiency, lacks the naturalness and the explanation capability of the symbolic component. This is so, because pre-eminence is given to the connectionist framework.

On the other hand, connectionist expert systems are integrated systems that represent relationships between concepts, considered as nodes in a neural network. Weights are set in a way that makes the network infer correctly. The system in (Gallant 1993) is a popular such system, whose inference engine is called MACIE. Two characteristics of MACIE are: its ability to reason from partial data and its ability to provide explanations in the form of if-then rules. However, its inference process lacks naturalness. Again, this is due to the connectionist approach.

To improve the performance of connectionist expert systems, the "recency inference engine" and its corresponding explanation algorithm are introduced in (Ghalwash 1998). In order to assess its performance, which is better than MACIE, the 'convergence rate' is used, which is based on the number of known and necessary/required inputs. However, this measure does not take into account the internal number of computations made, which for large networks may be of importance.

In this paper, we present a hybrid inference engine and its associated explanation mechanism. The inference engine is related to *neurules*, a hybrid rule-based representation scheme integrating symbolic rules with neurocomputing, which gives pre-eminence to the symbolic component (Hatzilygeroudis and Prentzas 2000a, 2000b). Apart from naturalness, experimental results demonstrate an improvement to the efficiency of the inference compared to those in (Gallant 1993) and (Ghalwash 1998).

The structure of the paper is as follows. Section 2 presents neurules and Section3 the hybrid inference process introduced here with an example. In Section 4, the explanation mechanism is presented. Finally, Section 5 presents some experimental results and concludes.

2. Neurules

2.1 Syntax and Semantics

Neurules are a kind of hybrid rules. The form of a neurule is depicted in Fig.1a. Each condition C_i is assigned a number sf_i, called its *significance factor*. Moreover, each rule itself is assigned a number sf_0, called its *bias factor*. Internally, each neurule is considered as an adaline unit (Fig.1b). The *inputs* C_i (i=1,...,n) of the unit are the *conditions* of the rule. The weights of the unit correspond to the significance factors of the neurule and its bias is the bias factor of the neurule. Each input takes a value from the following set of discrete values: [1 (true), -1 (false), 0 (unknown)]. The *output* D, which corresponds to the *conclusion* (decision) of the rule, is calculated via the formulas:

$$D = f(a),\ a = sf_0 + \sum_{i=1}^{n} sf_i C_i \qquad (1)$$

* Alphabetical order.

where **a** is the *activation value* and *f(x)* the *activation function*, a threshold function:

$$f(\mathbf{a}) = \begin{cases} 1 & \text{if } a \geq 0 \\ -1 & \text{otherwise} \end{cases} \quad (2)$$

Hence, the output can take one of two values, '-1' and '1', representing failure and success of the rule respectively.

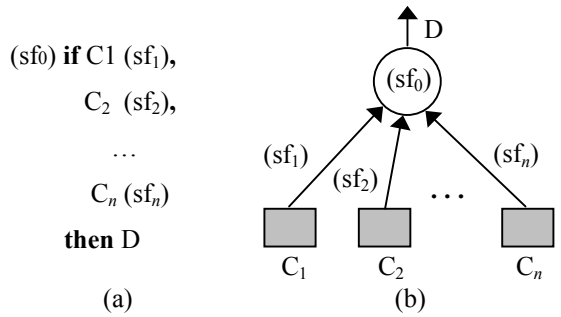

Fig.1 (a) Form of a neurule (b) corresponding adaline unit

The general syntax of a condition C_i and the conclusion D :

<condition>::= <variable> <l-predicate> <value>
<conclusion>::= <variable> <r-predicate> <value>

where <variable> denotes a *variable*, that is a symbol representing a concept in the domain, e.g. 'sex', 'pain' etc, in a medical domain. A variable in a condition can be either an *input variable* or an *intermediate variable*, whereas a variable in a conclusion can be either an *intermediate* or an *output variable* or both. An input variable takes values from the user (input data), whereas intermediate and output variables take values through inference, since they represent intermediate and final conclusions respectively. <l-predicate> and <r-predicate> are one of {is, isnot}. <value> denotes a value. It can be a *symbol* or a *number*. Neurules are distinguished in *intermediate* and *output rules*, depending on whether their conclusions contain intermediate or output variables respectively.

2.2 An Example Neurule Base

Neurules are constructed either directly, from empirical data, or by converting symbolic rules. In both cases, each neurule is individually trained via the LMS algorithm. Normally, for each possible conclusion one neurule is produced. However, in case of inseparability in the training set, where special techniques are used (Hatzilygeroudis and Prentzas 2000c, 2001), more than one neurule are produced with the same conclusion. The conditions of a neurule are sorted so, that $|sf_1| \geq |sf_2| \geq \ldots \geq |sf_n|$.

To illustrate the functionalities of our system, we use as an example the one presented in (Gallant 1993). It contains training data dealing with acute theoretical diseases of the sarcophagus. There are six symptoms (Swollen feet, Red ears, Hair loss, Dizziness, Sensitive aretha, Placibin allergy), two diseases (Supercilliosis, Namastosis), whose diagnoses are based on the symptoms and three possible treatments (Placibin, Biramibio, Posiboost). Also, dependency information is provided. We used the dependency information to construct the initial neurules and the training data provided to train them. The produced knowledge base, which contains six neurules (DR1-DR6), is illustrated in Fig.2.

DR1: (-0.4) **if** HairLoss is true (3.6),
SwollenFeet is true (3.6),
RedEars is true (-0.8)
then Disease is Supercilliosis

DR2: (1.4) **if** Dizziness is true (4.6),
SensitiveAretha is true (1.8),
HairLoss is true (1.8)
then Disease is Namastosis

DR3: (-2.2) **if** PlacibinAllergy is true (-5.4),
Disease is Supercilliosis (4.6)
Disease is Namastosis (1.8),
then Treatment is Placibin

DR4: (-4.0) **if** HairLoss is true (-3.6),
Disease is Namastosis (3.6),
Disease is Supercilliosis (2.8)
then Treatment is Biramibio

DR5: (-2.2) **if** Treatment is Biramibio (-2.6),
Treatment is Placibin (1.8)
then Treatment is Posiboost

DR6: (-2.2) **if** Treatment is Placibin (-1.8),
Treatment is Biramibio (1.0)
then Treatment is Posiboost

Fig.2 An example neurule base.

3. The Hybrid Inference Engine

3.1 The Process

The *hybrid inference engine* implements the way neurules co-operate to reach a conclusion, which is based on the 'firing potential', a measurement of the firing tendency of a neurule, which is similar to the 'convergence ratio' introduced in (Ghalwash 1998). The basic idea is: in each inference step, consider first the neurule with the largest firing potential, because it is the most likely to fire.

Normaly, the output of a neurule is computed according to Eq. (1). However, it is possible to deduce the output of a neurule without knowing the values of all of its conditions. To achieve this, we define for each neurule the *known sum (kn-sum)* and the *remaining sum (rem-sum)* as follows:

$$kn - sum = sf_0 + \sum_{c_i \in E} sf_i C_i \quad (3)$$

$$rem - sum = \sum_{c_i \in U} |\, sf_i \,| \quad (4)$$

where E is the set of its evaluated conditions, U the set of its unevaluated conditions and C_i is the value of the i^{th} condition. So, 'known-sum' is the weighted sum of the values of the already known (i.e. evaluated) conditions (inputs) of the corresponding neurule and 'rem-sum' represents the largest possible weighted sum of the remaining (i.e. unevaluated) conditions of the neurule. If $|kn\text{-}sum| > rem\text{-}sum$, for a certain neurule, then evaluation of its conditions can stop, because its output can be deduced regardless of the values of the remaining unevaluated conditions. So, we define the *firing potential (fp)* of a neurule:

$$fp = \frac{|kn\text{-}sum|}{rem\text{-}sum} \qquad (4)$$

which is an estimate of its tendency to make its output '±1'. Whenever $fp > 1$, the rule evaluates to '1' (true), if $kn\text{-}sum > 0$ or to '-1' (false), if $kn\text{-}sum < 0$. In the first case, we say that the neurule is *fired*, whereas in the second that it is *blocked*. Notice that *fp* has meaning only if $rem\text{-}sum \neq 0$. If $rem\text{-}sum = 0$, all the conditions have been evaluated and its output is evaluated according to *kn-sum*.

The inference process is as follows, where the 'working memory' (WM) is a place for keeping data.

1. Initially, set the *fp*s of all the neurules to their bias factors. If there is initial input data in the WM, find all the affected neurules and update their *fp*s; they become the *participating neurules*. Otherwise, regard all neurules as participating.
2. Do the following,
   2.1 If there is a participating neurule with ($fp > 1$ or $rem\text{-}sum = 0$) then if ($kn\text{-}sum > 0$), mark the rule as *fired* and its conclusion as 'true' and put it in the WM, otherwise ($kn\text{-}sum < 0$), mark the rule as *blocked* and if there is no other unevaluated neurule with the same conclusion, mark its conclusion as 'false' and put it in the WM.
   2.2 Remove the above evaluated neurule from the participating rules.
   2.3 If the condition put in the WM is an intermediate one, find the affected neurules, put them in the participating neurules and update their *fp*s.

   until there is no participating neurule with $fp > 1$ or $rem\text{-}sum = 0$.
3. While there are participating neurules do,
   3.1 From the participating neurules select the one with the maximum *fp*. If there are no participating neurules, select an unevaluated one with the maximum *fp*.
   3.2 Consider the first unevaluated condition of the selected neurule. If it contains an input variable, ask the user for its value and put it in the WM. If it contains an intermediate variable instead, find an unevaluated neurule with the maximum *fp* that contains the variable in its conclusion and execute this step recursively taking this neurule as the selected.
   3.3 Clear participating rules. According to the input data, find all the affected neurules, update their *fp*s and put them in the participating neurules.
   3.4 (the same as step 2).
4. If there are no conclusions in the WM containing output variables, stop (failure). Otherwise, display the conclusions and stop (success).

A neurule is *evaluated*, if it is fired or blocked, otherwise it is *unevaluated*. Also, *affected neurules* are those unevaluated neurules containing at least a condition with the same variable as that of the conclusion put in the WM.

3.2 An Example Inference

In this section, a hand tracing of an example inference from the neurule base in Fig. 2 is presented. Notice that DR1 and DR2 are intermediate neurules, DR3 and DR4 are simultaneously intermediate and output neurules, whereas DR5 and DR6 are output neurules. Initially, the *fp* of each neurule is set to its bias factor.

Step 1
WM: {'HairLoss is true' (TRUE)} (Initial data)
Affected neurules: [DR1, DR2, DR4]
Updated *fp*s: |3.2/4.4| = 0.73 (DR1), |3.2/6.4| = 0.5 (DR2), |-7.6/6.4| = 1.19 (DR4)
Partcipating neurules: [DR1, DR2, DR4]

Step 2
Step 2.1 (DR4 has $fp > 1$ and $kn\text{-}sum < 0$)
Blocked neurules: [DR4]
WM: {'HairLoss is true' (TRUE), 'Treatment is Biramibio' (FALSE)}
Step 2.2
Participating neurules: [DR1, DR2]
Step 2.3
Affected neurules: [DR5, DR6]
Partcipating neurules: [DR5, DR6, DR1, DR2]
Updated *fp*s: |0.4/1.8| = 0.22 (DR5), |-3.2/1.8| = 1.78 (DR6)

Step 2
Step 2.1 (DR6 has $fp > 1$ and $kn\text{-}sum < 0$).
Blocked neurules: [DR6, DR4]
Step 2.2
Participating neurules: [DR5, DR1, DR2]

Step 3
Step 3.1
Selected neurule: DR1
Step 3.2
User data: 'SwollenFeet is true' (FALSE)
WM: {'HairLoss is true' (TRUE), 'Treatment is Biramibio' (FALSE), 'SwollenFeet is true' (FALSE)}
Step 3.3
Participating neurules: [ ]
Affected neurules: [DR1]
Updated *fp*s: |-0.4/0.8| = 0.5 (DR1)
Participating neurules: [DR1]
Step 3.4 (no effect)

Step 3.1
Selected neurule: DR1

Step 3.2
User data: 'RedEars is true' (FALSE)
WM: {'HairLoss is true' (TRUE),
'Treatment is Biramibio' (FALSE),
'SwollenFeet is true' (FALSE),
'RedEars is true' (FALSE)}

Step 3.3
Participating neurules: [ ]
Affected neurules: [DR1]
Updated *fp*s: *kn-sum* = 0.4, *rem-sum* = 0
Participating neurules: [DR1]

Step 3.4 (2)

Step 3.4.1 (2.1)
Fired neurules: [DR1]
WM: {'HairLoss is true' (TRUE),
'Treatment is Biramibio' (FALSE),
'SwollenFeet is true' (FALSE),
'RedEars is true' (FALSE),
'Disease is Supercilliosis' (TRUE)}

Step 3.4.2 (2.2)
Participating neurules: [ ]

Step 3.4.3 (2.3)
Affected neurules: [DR3] (DR4 has been evaluated)
Participating neurules: [DR3]
Updated *fp*s: |2.4/7.2| = 0.33 (DR3)
(no neurule with *fp* > 1)

Step 3.1
Selected neurule: DR3

Step 3.2
User data: 'PlacibinAllergy is true' (FALSE)
WM: {'HairLoss is true' (TRUE),
'Treatment is Biramibio' (FALSE),
'SwollenFeet is true' (FALSE),
'RedEars is true' (FALSE),
'Disease is Supercilliosis' (TRUE),
'PlacibinAllergy is true' (FALSE)}

Step 3.3
Affected neurules: [DR3]
Updated *fp*s: |7.8/1.8| = 4.33 (DR3)
Participating neurules: [DR3]

Step 3.4 (2)

Step 3.4.1 (2.1)
Fired neurules: [DR1, DR3]
WM: {'HairLoss is true' (TRUE),
'Treatment is Biramibio' (FALSE),
'SwollenFeet is true' (FALSE),
'RedEars is true' (FALSE),
'Disease is Supercilliosis' (TRUE),
'PlacibinAllergy is true' (FALSE),
'Treatment is Placibin' (TRUE)}

Step 3.4.2 (2.2)
Participating neurules: [ ]

Step 3.4.3 (2.3)
Affected neurules: [DR5]
Participating neurules: [DR5]
Updated *fp*s: *kn-sum* = 2.2, *rem-sum* = 0

Step 3.4.1 (2.1)
Fired neurules: [DR5, DR3, DR1]
WM: {'HairLoss is true' (TRUE),
'Treatment is Biramibio' (FALSE),
'SwollenFeet is true' (FALSE),
'RedEars is true' (FALSE),
'Disease is Supercilliosis' (TRUE),
'PlacibinAllergy is true' (FALSE),
'Treatment is Placibin' (TRUE),
'Treatment is Posiboost' (TRUE)}

Step 3.4.2 (2.2)
Participating neurules: [ ]

Step 3.4.3 (2.3)
Participating neurules: [ ]

So, we have the following final conclusions:

Output data: 'Treatment is Placibin', 'Treatment is Posiboost'

4. The Explanation Mechanism

The explanation mechanism justifies inferences by producing a set of simple if-then rules, explaining how the conclusions were reached. The conclusions of the explanation rules contain the inferred output variables. Their conditions contain a subset of the input and intermediate variables participating in drawing the conclusions, that is those variables whose values were either given by the user or inferred during the inference process, possibly with changes to their predicates. More specifically, the conditions in the explanation rules are the ones with the most positive contribution in producing the output of the corresponding neurule. We call them *positive conditions*, whereas the rest *negative conditions*.

In case a neurule's output evaluates to '1', the positive conditions are either the ones evaluated to true ('1') and having a positive significance factor or the ones evaluated to false ('-1') and having a negative significance factor. In case a neurule's output evaluates to '-1', the negative conditions are either the ones evaluated to true ('1') and having a negative significance factor or the ones evaluated to false ('-1') and having a positive significance factor. Conditions that are unknown or negative are not included in explanation rules. Furthermore, some of the positive conditions may be also not included, based on the fact that they are not necessary. The unnecessary positive conditions are the ones with the smallest absolute significance factors.

For each of the fired output neurules, the explanation mechanism generates an if-then rule whose conclusion is the neurule's conclusion and its conditions are the necessary positive conditions of the neurule. Possible changes are made to the predicates according to the values of the conditions (e.g. if a necessary positive condition is evaluated to false, its predicate is changed from 'is' to 'isnot' and vice versa). In addition, for each condition

containing an intermediate variable, an if-then rule is produced based on an evaluated neurule having that condition as its conclusion. This process recurses.

Table 1. The extracted explanation rules

EXR1 **if** HairLoss is true **then** Treatment isnot Biramibio	EXR2 **if** HairLoss is true, RedEars is false **then** Disease is Supercilliosis
EXR3 **if** PlacibinAllergy is true, Disease is Supercilliosis **then** Treatment is Placibin	EXR4 **if** Treatment isnot Biramibio, Treatment is Placibin **then** Treatment is Posiboost.

The explanation rules extracted for the example inference described in section 3.2 are shown in Table 1. In this case, there are two outputs, 'Treatment is Posiboost' and 'Treatment is Placibin', and the explanation mechanism provides explanations for them. It is easy then to produce a text explaining the inference.

5. Experimental Results and Conclusion

This section presents experimental results comparing the performance of our inference mechanism with that presented in (Gallant 1993) and (Ghalwash 1998). Our inference mechanism was applied to two neurule bases, directly created from two datasets (described below). The inference mechanisms in (Gallant 1993) and (Ghalwash 1998) were applied to two connectionist knowledge bases, created from the same datasets by the technique described in (Gallant 1993). Both connectionist knowledge bases are multilevel networks. The comparison is made in terms of the number of inputs asked by the system in order to draw conclusions (as suggested in (Ghalwash 1998)) and the number of the conditions/inputs visited for some kind of computation in drawing the conclusions.

The first dataset is that used in Section 3.2 and taken from (Gallant 1993). This dataset is incomplete. It consists of 8 input data patterns out of 64 possible. We ran the experiments with the 56 cases. The second dataset is taken from the machine learning ftp repository (see Dataset in the References) and involves a database for fitting contact lenses. This dataset is complete and contains 24 input patterns each consisting of four input and one output attribute (variable) which takes three possible values.

Table2. Experimental results

KB	*PATS*	*NEURULES*		*GALLANT*		*GHALWASH*	
		ASK	*VIS*	*ASK*	*VIS*	*ASK*	*VIS*
KB1	56	202	586	231	986	207	882
KB2	24	79	602	101	443	80	886

Table 2 depicts the results. Initially, the values of all variables were not known. KB1and KB2 are the two knowledge bases and PATS the training patterns. ASK and VIS denote 'asked inputs' and 'conditions visited'.

Table 2 shows that our inference engine performed quite better than Gallant's and slightly better than Ghalwash's as far as the number of asked inputs is concerned. Also, it did much better, on the average, than both the other systems as far as conditions visits are concerned. Although this is not significant for small knowledge bases, it may become important for very large ones.

On the other hand, due to the existence of intermediate cells in the other systems, the number of explanation rules produced by the explanation mechanisms in (Gallant 1993, Ghalwash 1998), to justify the same conclusions, are more than the ones produced by our explanation mechanism. This fact, besides the computational cost, raises an issue of comprehensibility as far as the user is concerned. The more explanation rules are presented to the user, the more confused he/she is.

So, experimental results show an improvement to the performance of the inference engine compared to pure connectionist approaches. Also, the explanation mechanism seems to produce shorter explanations.

Acknowledgments

This work was partially supported by the GSRT of Greece, Program ΠΕΝΕΔ-1999, Project No 99ΕΔ234.

References

Dataset, ftp:// ftp.ics.uci.edu / pub / machine – learning - databases/

Gallant, S.I. 1993. *Neural Network Learning and Expert Systems*, MIT Press.

Ghalwash, A. Z. 1998. A Recency Inference Engine for Connectionist Knowledge Bases, *Applied Intelligence*, 9:201-215.

Hatzilygeroudis, I. and Prentzas, J. 2000a. Neurules: Integrating Symbolic Rules and Neurocomputing, in Fotiades, D., and Nikolopoulos, S. eds. *Advances in Informatics*, 122-133. World Scientific.

Hatzilygeroudis, I. and Prentzas, J. 2000b. Neurules: Improving the Performance of Symbolic Rules. *International Journal on Artificial Intelligence Tools* 9(1):113-130.

Hatzilygeroudis, I. and Prentzas, J. 2000c. Producing Modular Hybrid Rule Bases for Expert Systems. In *Proceedings of the 13th International FLAIRS Conference*, 181-185. Orlando, FL.

Hatzilygeroudis, I. and Prentzas, J. 2001. Constructing Modular Hybrid Rule Bases for Expert Systems. *International Journal on Artificial Intelligence Tools* 10(1-2). Forthcoming.

Medsker, L. R. 1994 *Design and Development of Expert Systems and Neural Networks*. New York: Macmillan Publishing Company.

Tirri, H. 1995. Replacing the Pattern Matcher of an Expert System with a Neural Network. In Goonatilake, S. and Sukdev, K. eds. *Intelligent Hybrid Systems*. John Wiley & Sons.