629·8 (S)

# A Heuristic for General Rule Extraction from a Multilayer Perceptron

**Zhe Ma, Robert F Harrison**
Department of Automatic Control and Systems Engineering
The University of Sheffield
PO Box 600, Mappin Street, Sheffield S1 4DU, UK
E-mail: z.ma@shef.ac.uk, R.F.Harrison@shef.ac.uk

## Abstract

Rule extraction from Artificial Neural Networks (ANNs) is an essential step towards the integration of ANNs and Knowledge-based Systems (KBSs). Two central questions addressed in this paper are what is a suitable format embodying ANN knowledge correctly and efficiently; and how is the knowledge extracted.

A General Rule is defined in an efficient format to represent the knowledge from ANNs. General Rules are extracted from a trained Multilayer Perceptron (MLP). The inputs of the MLP correspond to the premises, and the outputs, to the conclusions of the rules. Two criteria are used to ascertain the significance of input components for the construction of rules.

The first criterion, the Potential Default Set (PDS) is drawn up from the weighted connections combined with the input/output correspondence of a training pattern. A subset of the inputs in the training pattern which is possibly redundant, is defined as the PDS.

The second criterion, the Feature Salient Degree (FSD) is computed by checking through the training pattern set. The FSD embodies the causal bond of the changes on each input bit and on each output bit.

The system using both PDS and FSD is demonstrated by application to typical logic problems such as AND, OR, XOR and by interpretation for unknown data. Clinical data have also been used to assess the performance of the method in the real-world. The rules derived here have been evaluated by a domain expert and are found to conform with his view of the problem.

The computational complexity is a third order polynomial of the problem size.

*Key Words*: Rule Extraction, Hybrid Knowledge-based System, Neural Network

**The University of Sheffield, Department of Automatic Control & Systems Engineering Research Report No 549**

## 1. The Role of Rule Extraction in Knowledge Engineering

Artificial Neural Networks (ANNs) and Knowledge-based Systems (KBSs) represent knowledge and intelligence processes at different levels [Fu 94]. Integration of the two types of system promises to overcome their individual shortcomings. ANNs can automatically learn emergent properties from the original domain data, but it is difficult to show explicit knowledge of what it has learned. Conversely, most KBSs use rules of inference based on high level knowledge representations sympathetic to the human, but suffer the knowledge acquisition bottleneck.

Successful integration of ANNs and KBSs may lead to completely automatic processing throughout the life-cycle of knowledge engineering. Extraction of rules from the ANN for use by the KBS is the essential step in the integration of the two systems.

A desirable characteristic of rule extraction is that the rules from the ANN should be: capable of interpretation by the KBS; accurate and complete with respect to what the ANN has learned; and efficient for the KBS to use. We present in this paper a method of extracting rules from a Multilayer Perceptron (MLP) feedforward neural network which fulfils the requirements.

The paper is organised as follows. Section 2 reviews some related work on Rule Extraction from ANNs. Section 3 explains the advantages of the format for the General Rules, also included is a discussion of the terminology used. The extraction method for the General Rules is given in Section 4. Section 5 reports on some experiments and Section 6 further discusses the method and summarises the central issues in this paper.

## 2. Review of Previous Work

Rule extraction from neural networks can be categorised into Black-Box [Sait 90] or Open-Box (White-Box) [Towe 90] approaches. The former ignores the internal structures of ANNs, and generates rules referring only to the correspondence of input and output values. The latter identifies rules according to observation of the internal connection structures of ANNs.

[Sait 90] extracts rules from a trained MLP based on correlation of inputs and outputs, as the MLP is fed with input patterns with bits switched. This leads to a combinatorial problem and some effort was expended to try to relieve this, such as by limiting the switching only to positive input bits among the training patterns, and controlling the number of input bits to be switched. [Towe 90], [Towe 93A], and [Towe 93B] explain the KBANN system, which first encodes the domain knowledge (rules) as the connections in an MLP, then extracts the refined rules from the trained network. In order to reduce combinations of the antecedents of the rules, KBANN also modifies the weights into a few uniform groups during a secondary training without changing the knowledge encoded, identical weights in a group can then be represented as an MofN relation. [Carp 93] shows extraction of fuzzy rules from a self-organising supervised learning neural network -- fuzzy ARTMAP, by two techniques: pruning the nodes which correspond to the rules to be extracted, and quantifying learned weights. [Ride 94] develops a collection of methods to generate hierarchical "if then" rules from a multilayer Class-Entropy Minimisation Network. The rule extraction procedure comprises partitioning-pruning-retraining cycles. The rules are extracted from every connection layer respectively. As a consequence they are in a format of a cascaded embedding of the form "if-then-else...". [Boch 90] uses a general formula called the validity domain to ease the generation of rules from an MLP. [Died 90] demonstrates a connectionist semantic network having an explicit conceptual hierarchy as an explanation component to answer "how" questions. [Maho 93] describes the RAPTURE system, which uses two subsystems: a modified MLP to refine the certainty factors of a MYCIN style rule, and a symbolic learning method with the ID3 information gain heuristic for addition of new rules. [Fu 93] presents a system named

KBCNN which constructs and revises a neural network according to the rules encoding the initial domain knowledge. When the network performance gets stuck during training, new hidden units are added to different layers in order to generate new concepts. Rules are generated directly from the weighted connections in the trained network. [McMi 91] explains a system called RuleNet which extracts rules from a neural network and re-encodes them, in an iterative projection process.

The great success of the MLP in machine learning and the fact that it is most widely adopted ANN, most attempts at rule extraction from ANNs have been made on this architecture or on its variants. However, rule extraction from MLPs has proved to be difficult owing to the MLP's behaviour of mingling multiple nonlinear parameters and the mutual dependence of each layer.

In the Black-Box approach, researchers face the combinatorial obstacle. The number of situations to test is exponentially increasing with the domain size. Limits on the test number lead to the likelihood of missing some features possibly crucial to the domain.The limitations of the Open-Box approach include: as a special variety of the MLP architecture is designed, the generality and learning capability are guaranteed; network pruning requires retraining which can be very time consuming and unreliable; only certain linear features have been identified, therefore a non-linear domain requires a larger network.

## 3. General Rules: An Alternative Approach

Generalisation of the extracted rules is very important. If most I/O correspondence in an ANN is directly translated into rules, a KBS will suffer more severe combinatorial problems than the ANN. This is because symbolic reasoning usually requires more computation than subsymbolic inference about problems of the same size. Many practical applications would be meaningless to the human if many large rules were produced without generalisation.

Before going further, we explain some terminology to be used in this paper.

***MLP*** The Multilayer Perceptron here is assumed to have only one layer of hidden units. Complete weighted connections are used for any adjacent layers. The input, hidden and output layers of units are denoted as $\{I_i\}$, $\{H_h\}$ and $\{O_o\}$ respectively. The two layers of weight connections from input to hidden layer and from hidden to output layer are $W_1=\{w_{hi}\}$ and $W_2=\{w_{oh}\}$ respectively. Binary domains only are of concern here, *i.e.* the input patterns are vectors of 0 and 1. A trained MLP will give pseudo-binary outputs too, *i.e.* the values are in the range either $[0, \Delta]$ or $[1-\Delta, 1]$, where the $\Delta$ is a pre-defined *error tolerance.*

***Sole Pattern*** A sole pattern is an input pattern, taken together with one of the output bits. The input pattern is selected from the training patterns. The output bits are either picked from the training patterns or computed by the trained MLP. For example, for an MLP having q output units, if the output bits are picked from the training pattern set, a single *training pattern* includes q sole patterns, each of which comprises the input pattern and a different output bit.

***Rule*** A rule to be extracted is in the form

$$\text{IF}(C_1, C_2,..., C_n) \text{ THEN } (R_j);$$

being explained as if all the ***premises*** (or conditions) $C_1$, $C_2$,... and $C_n$ are true, the ***conclusion*** $R_j$ is true. Both $C_i$ and $R_j$ are propositions, being either positive or negative (headed with a ~ if being negative). This form is equivalent to the Horn Clause Format. The premises of a rule are projected from the input bits of a sole pattern. The conclusion is obtained from the output bit of the sole pattern. A rule having a complete set of premises, *i.e.* the number of premises is the length of the input pattern, is a ***Special Rule***. A rule having an incomplete set of premises is a ***General Rule***. In a General Rule, the absent features are *not significant* in forming the conclusion, not false as some other systems imply.

A General Rule is equivalent to a set of special rules with all combinations of the values on those absent premises in the General Rule. The fewer the premises a rule has, the more general it is. The General Rule represents knowledge in a format of high level abstraction. It preserves the salient features necessary to hold the rule's property and ignores the trivial features. A few General Rules will be sufficient for description of a particular domain if some general qualities exist. Reasoning by General Rules is fast since only hard matching is needed. This allows the KBS efficiently to retrieve the rules, to maintain the rule base (including the consistency check), and to support knowledge acquisition. The General Rule enables the KBS more suitably to generalise knowledge to new situations since the General Rule may cover a region beyond that encountered by the system (see the example in Section 5.4).

General Rules provide system reliability, because they overlap in the problem domain, enabling the KBS to confirm a special situation by application of multiple rules. After the Rule Validation procedure, the remained General Rules are still able to cover the most (if not the whole) problem domain.

General Rules are analogous to the human expression of knowledge, and human knowledge formation shares similar advantages to those stated above.

As the later examples show, the size of rule base formed by General Rules is not necessarily smaller than by using other methods. A certain amount of redundancy is beneficial to generalisation and system reliability. The size of the rule base formed of General Rules depends on the domain size, increasing at a polynomial rate, and on the information density of the training data.

## 4. Extracting the General Rules

Given a sole pattern, how can we find the smallest input subset which is sufficient to retain the value of the output bit regardless what values are given to the rest of the input units? Instead of exhaustive tests by switching all combinations of the input values, which demands a computation of $O(2^N)$ for an N bit input set, our method requires a polynomial computation of the domain size. The method takes advantage of both the Black-Box and the Open-Box approaches.

From the Open-Box approach, a "*contribution*" relationship is obtained from the input units to the output units using the weights of the trained MLP. The input bits in a sole pattern are grouped into supporters and opponents of the output by this relationship. The opponents are possibly redundant.

From the Black-Box approach, a Feature Salient Degree (FSD) is computed for every input bit in a sole pattern, reflecting the possibility of a change to the output bit if the value of the input bit is switched.

Combining the two criteria above, most bits in a sole pattern will be deemed redundant or not. Generalised rules are collected by dropping the redundant bits. Further compression is taken by the merging of special rules into a general one. Rule Validation is used for information sparse or noisy situations.

In section 4.1, we explicate the Open-Box approach. Section 4.2 explains how to compute the FSD in the training set. Both results are used in the algorithm for rule generation in Section 4.3.

### 4.1 Potential Default Set

In a trained MLP, an input unit $I_i$ links an output unit $O_o$ via the weighted connections with all hidden units. The link strength via one hidden unit $H_h$ is $w_{hi} \cdot w_{oh}$. The summed link strength of the $I_i$ to the $O_o$ is $L_{oi} = \sum_h w_{hi} \cdot w_{oh}$.

The overall link strengths between the input units and the output ones are simply the product of the weight matrices $L = W_2 \times W_1$.

The matrix $L$ is explained as follows. If element $L_{oi}$ is positive, the input unit $I_i$ generally gives positive contribution to the output unit $O_o$ owing to the increasing nature of the logistic function. In other words, if $I_i$ is switched from 0 to 1 in the input set which is in turn fed forward through the MLP, $O_o$ generally receives an incremental activation from $I_i$. If $I_i$ is switched from 1 to 0, $O_o$ generally receives a decrement in activation from $I_i$. In the case when $L_{o,i}<0$, the situations are reversed. Focusing on a sole pattern $\{\{I_i\}, O_o\}$, assuming that $I_i$ and $O_o$ denote an input bit and the output bit with particular values[*] and a row $L_o=\{L_{oi}\}$ of the matrix $L$ corresponds to the $O_o$, let us define the following sets:

$$Z_0=\{I_i \mid I_i=1\} \qquad N_0=\{I_i \mid I_i=0\}$$
$$Z_1=\{L_{oi} \mid L_{oi} \geq 0\} \qquad N_1=\{L_{oi} \mid L_{oi} < 0\}$$

If the $O_o$ is in $[1-\Delta, 1]$, those inputs in $Z_0 \cap N_1$ or in $N_0 \cap Z_1$ have given negative activation contributions to $O_o$. Switching them may change their contributions to be positive. That may not change the output $O_o$ much because $O_o$ can only increase in the range $[1-\Delta, 1]$. Conversely, if $O_o$ is in $[0, \Delta]$ the inputs in $Z_0 \cap Z_1$ or in $N_0 \cap N_1$ have given positive contributions, and the $O_o$ may not be changed much by switching their values. We name such sets the ***Potential Default Sets,*** or ***PDSs***:

$$(Z_0 \cap N_1) \cup (N_0 \cap Z_1) \quad \text{if } O_o=1$$
$$(Z_0 \cap Z_1) \cup (N_0 \cap N_1) \quad \text{if } O_o=0$$

If any $I_i$ makes no significant change to an output by switching its value, it can be absent in the premises of the rule to be generated. The inputs in the PDS are those candidates which may be absent from the rule to be extracted without losing the property implied from the sole pattern.

The size of the PDS is statistically half of that of the input pattern. Hence the dimensionality of the test space on the input values may be reduced by half.

Although the PDS reflects the statistical properties of the trained MLP and can be correctly applied to extract the General Rules for some linear domains such as AND and OR problems, it is not sufficient for nonlinear domains such as two bit parity (XOR). The Feature Salient Degree introduced in the next section aims to overcome the linear limitation.

## 4.2 Feature Salient Degree

Concerning the sole patterns $\{P_j\}$ in respect to the ***same output unit***, we first define the quantity of ***feature salient degree (fsd)*** to an input bit $I_{ji}$ in a sole pattern $P_j$:

$$fsd_{ji} = \sum_{\{k \mid (j \neq k,\, o_j \neq o_k,\, I_{ji} \neq I_{ki})\}} dist(P_j, P_k)^{-2}$$

where $I_{ji}$ and $I_{ki}$ are the *ith* input values respectively in sole patterns $P_j$ and $P_k$, $O_j$ and $O_k$ are the output values involved in $P_j$ and $P_k$. The function $\mathrm{dist}(P_j,P_k)$ is the distance of the input sets in $P_j$ and $P_k$.

The *fsd* is a measure of the amount of information carried by the input units with respect to

[*] The $O_o$ could be the output value as $\{I_i\}$ are fed to the MLP. This makes a difference when the MLP does not learn the training set 100% accurately. See Section 6 for discussion.

the training data. It represents the correlation of the changes on an input bit with a particular value and an output bit with a particular value. It also reflects the distinct effect of the input on the output. Here, it estimates the possibility of a change of the output value when the input bit is switched.

For a sole pattern $P_j$, the more sole patterns having different output values from that of $P_j$, the higher the *fsd* of its input units. Between two sole patterns $P_j$ and $P_k$ with different output values, the smaller the dist($P_j$,$P_k$) is, and the higher the contributions to their *fsds*.

As the *fsd* defined above is variable relating to the input size and the training set size, a more general Feature Salient Degree ***FSD*** which is less affected by those sizes is defined as follows.

The FSD of a problem with N input bits is the FSD defined above divided by the FSD of the N-bit parity problem. The parity problem is highly non-linear and know to be hard for neural networks to learn. Each bit in each sole pattern carries the same high amount of information. The FSD of the N bit parity problem is

$$fsd0 = \sum_{i=0}^{\lfloor N/2 \rfloor} \binom{2i}{N-1} \cdot (2i+1)^{-2}$$

where $\binom{m}{n}$ indicates m combinations from n.

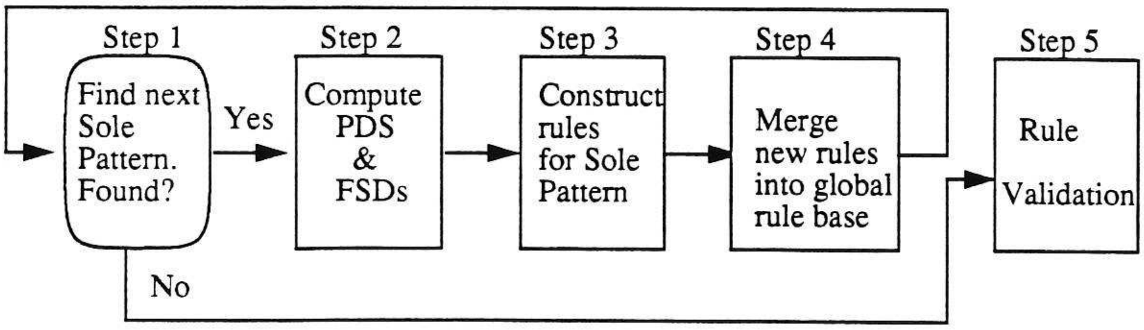
The FSD of an N input bit problem is

$$FSD_{ji} = fsd_{ji}/fsd0$$

We will use the FSD instead of the *fsd* in what follows.

## 4.3 Rule Generation Algorithm

The flow chart next shows the algorithm of rule extraction from an MLP with N inputs and P outputs on M training patterns.



Step 1
Find next Sole Pattern. Found?
Yes
No
Step 2
Compute PDS & FSDs
Step 3
Construct rules for Sole Pattern
Step 4
Merge new rules into global rule base
Step 5
Rule Validation

**Rule Generation Algorithm**

Step 1 and Step 2 are as described in previous Sections.

Step 3. This is the kennel procedure. A threshold $T$ on the FSD is defined in advance.

i) For every input bit $I_i$

if its FSD>$T$, $I_i$ is reserved for a premise;

if the FSD<$T$ and $I_i \in$ PDS, $I_i$ is absent;

ii) For all other $I_i$s from i) (FSD<$T$ and $I_i \notin$ PDS), select all subsets of them so that the summed FSDs of the subset >$T$.

iii) Each subset obtained in ii) combining with all reserved $I_i$ selected in i), taken together with the output bit of the sole pattern, are projected into a General Rule. A list of rules is therefore constructed.

Step 4. This deals with two sets of rules, the new list and the global rule base GRB. For a rule

R1 in the new list, do one of the operations in following cases:

i) *Special*: if R1 is special to any rule in the GRB, ignore R1.

ii) *General*: if R1 is more general than some rules in the GRB, delete all rules in the GRB special to R1, and insert R1 into the GRB.

iii) *Generalisable*: if there is a rule in the GRB which has the same reserved input bits as R1, but there is one bit having different value with that of R1, generalise the R1 (i.e.eliminate the different bit from R1). Then do as ii)

iv) *New*: otherwise R1 is new in the GRB, insert it into the GRB.

Step 5. Check each rule R2 in the global rule base through the training set:

For each sole pattern $P_k$:

If R2 is more general than $P_k$, feed input pattern to the MLP, check the result $O_j$ at the output bit corresponding to $P_k$ and the output value $O_k$ in $P_k$:

i) if $O_j = O_k$ (within the error tolerance), the Correct Occurrence Count plus 1;

ii) otherwise, Error Occurrence Count plus 1.

If Correct Occurrence Count > (Error Occurrence Count x 1.5), the rule is valid.

The threshold *T* for the FSD can be set around 1. The higher the threshold is, the more general rules will be generated. A suggested range for the threshold *T* is [0.8, 1.2].

When a General Rule is compared with a sole pattern, there are only two situations: the rule is more general than the sole pattern, or the two are incomparable. The premises of the rule are a subset of the input pattern of the sole pattern in the former case.

The valid rules can be applied to an input pattern if they are more general than it. The majority of the rules applicable decides the output value reasoned with the rule base.

### 4.4 Complexity Analysis

Assume the trained MLP with N inputs, Q hidden units, and P outputs, there are M training patterns, i.e. M × P sole patterns. It is usual that N >> P and M >> N.

The PDS requires a computation in $\max(O(N\times M\times P), O(N\times Q\times P))$, the latter value is for the operation $W_2\times W_1$. The FSD requires $O(N\times M^2\times P^2)$ which is in a higher order than that of PDS. Therefore the Algorithm ***Rule_generation*** takes a computation in $O(N\times M^3\times P^3)$.

## 5. Experiments

Our algorithm has been successfully tested on many experiments, including some well known logical problems and one real-world pathology domain. Note, there is only one output unit in the examples, the Sole Patterns are the same as training patterns.

### 5.1 Two Bit AND and Four Bit OR

This Section is to show how the PDS and FSD are computed and used.

The two bit AND problem is defined as in Figure a1.

| A | B, | C, |
|---|---|---|
| 0 | 1, | 0, |
| 1 | 0, | 0, |
| 0 | 0, | 0, |
| 1 | 1, | 1, |

**Figure a1. Sole Patterns for Two Bit AND**

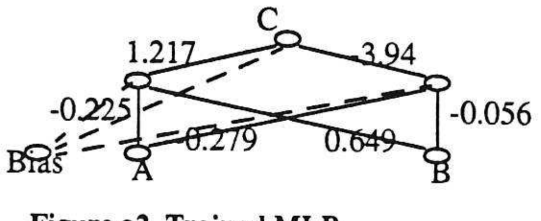


**Figure a2. Trained MLP for Two Bit AND**

The MLP is constructed with the trained weights in Figure a2.

$L=W_2 \times W_1 = [0.824, 1.01]$

$Z_0=\{A, B\}$, $N_0=\{\}$.

In first sole pattern (0 1,0), (i.e. A=0, B=1 and C=0)

$P_1=\{B\}$ and $N_1=\{A\}$. As the output is 0,

$PDS=(Z_0 \cap Z_1) \cup (N_0 \cap N_1)=\{B\}$.

The FSD for each bit in the sole pattern:

Sole pattern 1 has a different output only with sole pattern 3. The *fsds* for sole pattern 1 is

$fsd_{11}= \text{dist}(P_1, P_3)^{-2}=1$

$fsd_{12}=0$, since $I_{12}=I_{32}$ in sole pattern 1 and 3.

The *fsd0* for two bit XOR is 1 too. Therefore

$FSD_{11}=1$, $FSD_{12}=0$.

Set the threshold T=0.8. A=0 is reserved as a premise since $FSD_{11} > T$. The B is absent since $B \in PDS$ and $FSD_{12}<T$. The extracted rule from sole pattern 1 is

IF(~A) THEN (~C).

Table 1 lists the results for all the four sole patterns in the domain.

**Table 1: Two Bit AND**

| Sol_Pat | PDS | FSD | Rules |
|---|---|---|---|
| 0 1, 0 | {B} | 1 0 | IF(~A) THEN (~C) |
| 1 0, 0 | {A} | 0 1 | IF(~B) THEN (~C) |
| 0 0, 0 | {A,B} | 0.34 0.34 | None |
| 1 1, 1 | {} | 1.4 1.4 | IF(A,B) THEN (C) |

Only PDS or FSD alone is sufficient to decide the premise selection in this example. This is true in linear domains.

Similarly and more complex, the four bit OR problem is extracted as a set of rules:

IF (~B, ~C, ~D) THEN (~E); IF (~A, ~C, ~D) THEN (~E); IF (~A, ~B, ~D) THEN (~E);
IF (~A, ~B, ~C) THEN (~E); IF (A, B) THEN (E); IF (A, C) THEN (E);
IF (B, C) THEN (E); IF (A, D) THEN (E); IF (B, D) THEN (E);
IF (C, D) THEN (E);

which says that if any two of A, B, C and D are true then E is true; if any three of them are false, then E is false. We can change this easily to the form MofN [Towl 90]. There seems however no benefit to do so in automatic reasoning.

There is some redundancy in this representation. For example, those rules for conclusions E and those for ~E are complementary to each other. Each set can be absent as the default. In addition, there are non-empty intersections to most pairs of the rules for E (~E). For example, each rule for E includes four special rules, the last two rules

IF (B, D) THEN (E); IF (C, D) THEN (E);

have an intersection set of

IF (A, B, C, D) THEN (E); IF (~A, B, C, D) THEN (E);

There would be no difficulty in using the representation without redundancy in this case. Our current representation however possesses the advantage of being easily understandable and efficient for knowledge acquisition.

## 5.2 Two Bit XOR

XOR is the two bit parity problem. As a benchmark of FSD calculation, the FSD of every bit is 1. The PDS here is not sufficient for premise selection. The data for XOR is in Table 2.

**Table 2: Two Bit XOR**

| Sol_Pat | PDS | FSD | Rules |
|---|---|---|---|
| 0 1, 1 | {A} | 1 1 | IF(~A,B) THEN (C) |
| 1 0, 1 | {B} | 1 1 | IF(A,~B) THEN (C) |
| 1 1, 0 | {A,B} | 1 1 | IF(A, B) THEN (~C). |
| 0 0, 0 | {} | 1 1 | IF(~A,~B) THEN (~C) |

## 5.3 Asymmetric Domain

Here is another problem with four inputs, A, B, C, D and one output E. E is true if A is false and any other two inputs are true. If A is true, only all B, C and D being true can conclude E to be true. Otherwise, E is false. The sole patterns are:

**Table 3: Sole Patterns of Asymmetric Domain**

| Label | SP1 | SP2 | SP3 | SP4 | SP5 | SP6 | SP7 | SP8 | SP9 | SP10 | SP11 | SP12 | SP13 | SP14 | SP15 | SP16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| B | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| C | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| D | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| E | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |

L=[-1.40 1.15 1.33 1.15]

Threshold=0.8:

**Table 4: Results of Asymmetric Domain**

| Sol_pat | PDS | FSD | Rules |
|---|---|---|---|
| 0 0 0 0, 0 | {A} | 0.04 0.5 0.5 0.5 | IF (~B, ~C) THEN (~E); IF (~B, ~D) THEN (~E); IF (~C, ~D) THEN (~E); |
| 1 0 0 0. 0 | {} | 0.3 0.3 0.3 0.3 | None. There are 4 rules extracted, all are special to those above. |
| 0 1 0 0,0 | {A,B} | 0.08 0.08 1.1 1.1 | None. The extracted rule IF (~C, ~D) THEN (~E) appeared before. |
| ... | ... | ... | ...... |
| 1 1 1 1, 1 | {A} | 0.3 1.4 1.4 1.4 | IF (B, C, D) THEN (E); |

The extracted rules are

IF (~C, ~D) THEN (~E); IF (~B, ~D) THEN (~E); IF (A, ~D) THEN (~E); IF (~B, ~C) THEN (~E);
IF (A, ~C) THEN (~E); IF (A, ~B) THEN (~E); IF (~A, B, C) THEN (E); IF (~A, B, D) THEN (E);
IF (~A, C, D) THEN (E); IF (B, C, D) THEN (E);

just a necessary and sufficient set of General Rules to the conclusions ~E and E respectively.

## 5.4 Incomplete Training Set

Quality of learning may be assessed by reasoning on patterns not used in training. Most ANNs can only classify an unseen pattern intimate to its neighbours in terms of Euclidean distance. To a parity domain with an incomplete training pattern set, they can not correctly reason on the unseen patterns. The following is a four bit parity domain with an incomplete training set missing 5 patterns (shaded):

**Table 5: Sole Patterns of Incomplete 4 Bit Parity Domain**

| Label | SP1 | SP2 | SP3 | SP4 | SP5 | SP6 | SP7 | SP8 | SP9 | SP10 | SP11 | SP12 | SP13 | SP14 | SP15 | SP16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| B | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| C | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| D | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| E | 0 | 1 |  | 0 |  |  | 0 | 1 | 1 |  | 0 | 1 |  | 1 | 1 | 0 |

The trained MLP will reason on the whole sole pattern as shown in the following in a simplified format. The conclusions are rounded as integers if they fall in the tolerance range (0.1 here), including those patterns absent in training set:

**Table 6: Results by trained MLP for Incomplete 4 Bit Parity Domain**

| Label | SP1 | SP2 | SP3 | SP4 | SP5 | SP6 | SP7 | SP8 | SP9 | SP10 | SP11 | SP12 | SP13 | SP14 | SP15 | SP16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| B | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| C | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| D | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| E | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |

Only one untrained pattern "SP5" is correctly classified. The other four are misclassified.

The General Rules extracted from this trained MLP are

IF (~A, ~C, ~D) THEN (~E); IF (B, ~C, ~D) THEN (~E); IF (~A, B, ~D) THEN (~E);
IF (~A, B, ~C) THEN (~E); IF (A, B, C, D) THEN (~E); IF (A, ~B) THEN (E);
IF (A, C, ~D) THEN (E); IF (~B, D) THEN (E); IF (A, ~C, D) THEN (E);
IF (~B, C) THEN (E);

The rule validation procedure yields:

| | |
|---|---|
| Check rule IF (~A, ~C, ~D) THEN (~E); | Correct Occurrences: 1; Error Occurrences: 0 |
| Check rule IF (B, ~C, ~D) THEN (~E); | Correct Occurrences: 1; Error Occurrences: 0 |
| Check rule IF (~A, B, ~D) THEN (~E); | Correct Occurrences: 1; Error Occurrences: 0 |
| Check rule IF (~A, B, ~C) THEN (~E); | Correct Occurrences: 1; Error Occurrences: 0 |
| Check rule IF (A, B, C, D) THEN (~E); | Correct Occurrences: 1; Error Occurrences: 0 |
| Check rule IF (A, ~B) THEN (E); | Correct Occurrences: 2; Error Occurrences: 0 |
| Check rule IF (A, C, ~D) THEN (E); | Correct Occurrences: 1; Error Occurrences: 0 |
| Check rule IF (~B, D) THEN (E); | Correct Occurrences: 2; Error Occurrences: 0 |
| Check rule IF (A, ~C, D) THEN (E); | Correct Occurrences: 1; Error Occurrences: 0 |
| Check rule IF (~B, C) THEN (E); | Correct Occurrences: 1; Error Occurrences: 0 |
| Check rule IF (~A, C, D) THEN (E); | Correct Occurrences: 1; Error Occurrences: 0 |

All the General Rules are valid. Now apply the rules to the untrained patterns:

For input 0 1 0 0 in SP3, there are 4 rules are more general:

IF(~A, ~C, ~D)THEN(~E); IF(B, ~C, ~D)THEN(~E); IF(~A, B, ~D)THEN (~E); IF(~A, B, ~C)THEN(~E);

For 0 0 1 0 in SP5, 1 rule is more general than it:

IF (~B, C) THEN (E);

For 1 0 1 0 in SP6, 3 rules are more general:

IF (A, C, ~D) THEN (E); IF (~B, C) THEN (E); IF (A, ~B) THEN (E);

For 1 0 0 1 in SP10, 3 rules are more general:

IF (A, ~B) THEN (E); IF (~B, D) THEN (E); IF (A, ~C, D) THEN (E);

For 0 0 1 1 in SP13, 3 rules are more general:

IF (~B, D) THEN (E); IF (~B, C) THEN (E); IF (~A, C, D) THEN (E);

All conclusions are the same as given by the MLP.

## 5.5 A Clinical Data Set

A set of real-world data for the diagnosis of Breast Cancer is used in this Section. The data set consists of 413 patient records, each including ten binary-valued symptoms of breast tissue samples with a conclusion whether of a malignant or a benign lesion. Although the data set was claimed to have predictive value for the diagnosis task [Trot 91] [Koss 92], there are only 94 distinct input patterns and 12 of them correspond to conflict conclusions appearing at many places. Rule Validation is crucial in such noisy and redundant situation. Further details on this domain are introduced in [Down 94].

200 records were randomly selected for training and 38 General Rules were extracted. 32 valid rules remained after rule validation:

IF (~ICL, ~3D) THEN (Benign);
IF (~ICL, ~Nucleoli) THEN (Benign);
IF (~ICL, ~Size) THEN (Benign);
IF (~3D, Naked, Foamy, ~Necrotic) THEN (Benign);
IF (~3D, ~Nucleoli) THEN (Benign);
IF (~3D, ~Size) THEN (Benign);
IF (~3D, ~Necrotic, Apocrine) THEN (Benign);
IF (Naked, Foamy, ~Size) THEN (Benign);
IF (Naked, Pleo, ~Size) THEN (Benign);
IF (Naked, ~Size, ~Necrotic) THEN (Benign);
IF (Foamy, ~Nucleoli) THEN (Benign);
IF (Foamy, Pleo, ~Size) THEN (Benign);
IF (Foamy, ~Size, ~Necrotic) THEN (Benign);
IF (~Nucleoli, ~Size) THEN (Benign);
IF (~Nucleoli, ~Necrotic) THEN (Benign);
IF (~Nucleoli, Apocrine) THEN (Benign);
IF (Pleo, ~Size, ~Necrotic) THEN (Benign);
IF (~Size, Apocrine) THEN (Benign);
IF (ICL) THEN (Malignant);
IF (3D, ~Naked, Necrotic) THEN (Malignant);
IF (3D, ~Foamy) THEN (Malignant);
IF (3D, Nucleoli) THEN (Malignant);
IF (3D, Size) THEN (Malignant);
IF (~Naked, Nucleoli, Necrotic) THEN (Malignant);
IF (~Naked, Size, Necrotic) THEN (Malignant);
IF (~Foamy, Nucleoli) THEN (Malignant);
IF (~Foamy, Pleo) THEN (Malignant);
IF (~Foamy, Size) THEN (Malignant);
IF (Nucleoli, Pleo) THEN (Malignant);
IF (Nucleoli, Size) THEN (Malignant);
IF (Pleo, Size) THEN (Malignant);
IF (Size, Necrotic, ~Apocrine) THEN (Malignant).

These rules were used to check the remaining 213 records, resulting in 197 records correctly classified, 3 uncertain because there were 3 rules for and 2 rules against the conclusion in each case, 17 misclassified among which 9 were caused by contradictory appearances of the conclusions, i.e. there were different conclusions to identical input patterns. Disregarding the contradictory cases in the data, the correct rate was (213-17)/(213-9)=96%. The uncertain rate was 3/(213-9)=1.47%.

The valid rules were presented to an experienced consultant pathologist. He found that 29 of the 32 "fit in completely with [his] views" [Cros 94], and that only 3 (underlined) appeared anomalous at first sight. This was blamed on the appearance of "Pleo" in rules predicting benignancy. However, since this feature is always accompanied by "~Size", "It might be human misinterpretation..." in the assessment of Pleo(morphism) which is difficult for cells whose nuclei fall below the limits of "Size".

The rule extraction consumed 71.5 seconds CPU time on SUN Sparc 10 workstation. The rule validation and rule applications on the 213 records took 766 ms and 526 ms respectively.

# 6. Discussion and Conclusion

## 6.1 The Roles of PDS and FSD

In information dense situations, the FSD seems to provide determinant clues for feature selection. Nevertheless, the PDS is important since it gives statistical characterisation of the domain. For data in high-dimensional space the global property is likely to be more linear than non-linear, especially in information sparse situations.

## 6.2 Output Values in Sole Patterns

There are two ways to choose the output values for sole patterns, either taken from the training patterns or calculated by recalling the MLP as the input pattern are fed in. The results are identical if the MLP has learned the training set with 100% accuracy. However, if the MLP is not able to learn with 100% accuracy on the training set, either because there are contradictory sole patterns in the training set, or there are some complex properties beyond the MLP's capability (e.g. owing to a poor choice of architecture), all the results on the PDS, the FSD and therefore the General Rules are likely to be affected by the choice in the different ways. If the output values are chosen from the training set, the General Rules extracted reflect the properties of the training data set, which may complement the knowledge learned by the MLP with some features encoded in the training data set and not having been learned by the MLP. Contradictory appearances must be penalised in this case. If the output values are calculated by the MLP, the results inherit some advantages of the MLP such as generality and resistance to noise.

How to rate redundant appearances is another issue to be considered. This has not been done for the current algorithm.

## 6.3 FSD by Other Functions

The definition of the functional form for the FSD is not restricted to the one defined before. For instance, We have also used exponential functions for fsd and fsd0 in forming FDS:

$$fsd_{ji} = \sum_{\{k \mid (j \neq k,\, o_j \neq o_k,\, I_{ji} \neq I_{ki})\}} e^{-dist(P_j, P_k)} \quad \text{and} \quad fsd0 = \sum_{i=0}^{\lfloor N/2 \rfloor} \binom{2i}{N-1} \cdot e^{-(2i+1)}$$

All artificial examples in Section 5 give the same General Rules as using the previous definitions. There are slight differences on the Clinical Data domain: there were 48 General Rules extracted; 18 patterns were misclassified and 5 were uncertain among the remaining 213 patterns.

## 6.4 Confidence Degree for Rules

The General Rules are well formed for knowledge generalisation. They have covered all patterns in our experiments. We believe that if the occurrences observed in the rule validation procedure are taken into account as the ***Confidence Degree***, the General Rules can represent the domain knowledge more accurately, especially for deciding the cases where applicable rules have different conclusions. The Confidence Degree is more important if some previously unseen patterns are not covered by any General Rules, as in dealing with some information sparse domains, and some other technology such as soft-matching is to be employed.

## 6.5 Generality of the Method

The FSD is computed in the Black-Box style, and the PDS is a linear relationship on the MLP, both can be easily applied to most feed forward ANNs with different architectures.

## 6.6 Conclusion

The General Rule is a format representing only important data features, ignoring trivial ones. This representation of knowledge provides the capabilities of generalisation, simplicity and efficiency in knowledge engineering. It may also be used for explanation.

The General Rules are extracted from the training patterns using two criteria:

•The FSD reveals the salient features which are encoded in the training pattern set and are recognised by the MLP. It is the key criterion to decide which input values are redundant in the training context in order to generalise the extracted rules.

•The PDS reflects the statistical properties encoded in the trained MLP. Despite its linear limitation, it provides important clues for feature selection at a cost of some simple computation.

The heuristic described works well on some artificial problems and more importantly, on a real medical domain, is in agreement with medical opinion.

## References

[Boch 90] Laurent Bochereau and Paul Bourgine. *Rule extraction and validity Domain on a Multilayer Neural Network*. International Joint Conference on Neural Networks, 1990 Vol1 pp97-100

[Carp 93] Gail A. Carperter and Ah-Hwee Tan. *Rule Extraction, Fuzzy ARTMAP, and Medical Databases*. pp I-501-506

[Cros 94] Simon S Cross. Private Communication.

[Died 90] Joachim Diederich. *An Explanation Component for a Connectionist inference System*. EJCAI 90, pp222-227

[Down 94] J.Downs, R.F.Harrison, S.S. Cross. *A Neural Network Decision-Support Tool for the Diagnosis of Breast Cancer.* The University of Sheffield, Department of Automatic Control and Systems Engineering, Research Report No 548, 1994

[Fu 93] Fu, L.M. *Knowledge-based connectionism for revising domain theories*. IEEE Transactions on Systems, Man, and Cybernetics, 23(1), pp173-182

[Fu 94] Fu, L.M. *Neural Networks in Computer Intelligence*. 1994, McGraw-Hill

[Koss 92] L.G. Koss, (1992) *Diagnostic Cytology and its Histopathologic Basis* (4th Edition), pp 1293-1315.

[Maho 93] J. Jeffrey Mahoney, Raymond J. Mooney. *Combining Neural and symbolic Learning to Revise Probabilitic Rule base*. EJCAI 93, pp107-114

[McMi 91] Clayton McMillan, Michael C. Mozer, Paul Smolenskey. *Rule Induction through Integrated Symbolic and Subsymbolic Processing*. pp969-976

[Ride 94] S Ridella, G Speroni, P Trebino, R Zunino. *Class-Entropy Minimisation Networks for domain Analysis and Rule Extraction*. Neural computing and Applications, 1994 Springer-Verlag London Limited, pp 40-52

[Sait 90] K Saito, R Nakano, *Rule Extraction from Facts and Neural Networks*. International Neural Network Conference 1990, Vol1 PP379-382

[Towe 90] Geoffrey Towell, Jude W. Shavlik, M. O. Noordewier. *Refinement of Approximately correct Domain Theories by Knowledge-based Neural Networks*. Proceedings, 8th National Conference on AI, Boston, MA, pp861-866. AAAI Press

[Towe 93A] Geoffrey Towell, Jude W. Shavlik. *Extracting refined rules from knowledge-based neural networks*. Machine Learning, vol.13, no. 1,p.71-101

[Towe 93B] Geoffrey Towell, Jude W. Shavlik. *Interpretation of Artificial Neural Networks: Mapping Knowledge-Based Neural Networks into Rules*. EJCAI 93, pp977-984

[Towe 93C] Geoffrey Towell, Jude W. Shavlik. *Using Symbolic Learning to Improve Knowledge-Based Neural Networks*. AAAI 93

[Trot 91] P.A.Trott, (1991) *Aspiration Cytodiagnosis of the Breast*, Diagnostic Oncology, vol 1, pp79-87