A Study on Rule Extraction from Several Combined Neural Networks

Guido Bologna
Computer Science Centre
Rue General Dufour 24, Geneva 1211, Switzerland
Guido.Bologna@cui.unige.ch

Abstract

The problem of rule extraction from neural networks is NP-hard. This work presents a new technique to extract “if-then-else” rules from ensembles of DIMLP neural networks. Rules are extracted in polynomial time with respect to the dimensionality of the problem, the number of examples, and the size of the resulting network. Further, the degree of matching between extracted rules and neural network responses is 100%. Ensembles of DIMLP networks were trained on four data sets related to the public domain. Extracted rules were significantly more accurate than those extracted from C4.5 decision trees on average.

1 Introduction

Artificial neural networks are robust models used in classification problem domains. One major drawback is the fact that “knowledge” embedded therein is cryptically coded as a large number of weight and activation values. Golea showed that generating symbolic rules from neural networks is an NP-hard problem [12]. As a consequence, the difficulty to explain neural network responses is often one of the reasons limiting their use in practice. Recently, several authors proposed to explain neural network responses with symbolic rules given as: “if tests on antecedents are true then conclusion” [7, 11, 14, 15, 16, 20, 22, 23]. A taxonomy describing techniques used to extract symbolic rules from neural networks is proposed by Andrews et al. [1]. Basically, rule extraction methods are grouped into three methodologies: pedagogical, decompositional, and eclectic. In the pedagogical methodology, symbolic rules are generated according to an empirical analysis of input-output patterns. This is also called the black-box approach. In decompositional techniques, symbolic rules are determined by inspecting the weights at the level of each hidden neuron and each output neuron. Finally, the eclectic methodology both combines elements of the pedagogical and decompositional methodologies.

In a single Discretized Interpretable Multi-Layer Perceptron (DIMLP) [3, 4], rules are generated by determining discriminant hyper-plane frontiers. As a result, the degree of matching between network responses and rule classifications is 100%. Moreover, the computational complexity of the rule extraction
algorithm scales in polynomial time with the dimensionality of the problem, the number of examples, and the size of the network. Finally, continuous attributes do not need to be binary transformed as it is done in the majority of rule extraction techniques.

Recently, voting methods based on several combined classifiers have been investigated [17, 19]. Very often the accuracy of an ensemble of classifiers is higher than the accuracy of a single classifier. However, rule extraction at the level of the overall combination is still an open research problem [2, 19], though pedagogical rule extraction techniques could be applied [8]. This paper improves the current state of the art in rule extraction from neural networks by presenting an eclectic technique to generate “if-then-else” rules at the level of ensembles of DIMLP neural networks.

In the remaining sections, section 2 introduces the DIMLP model and the rule extraction technique, section 3 presents the bagging and arcing methods applied to ensembles of DIMLP networks, as well as the global rule extraction technique, section 4 presents the results carried out with three data sets of the public domain, followed in section 5 by the conclusion.

2 The DIMLP Model

The Discretized Interpretable Multi-Layer Perceptron [3, 4] has a special architecture that allow us to generate symbolic rules by determining discriminant frontiers. In the following paragraphs the key ideas behind rule extraction will be explained.

![Figure 1. A DIMLP network with two hidden layers.](image)

2.1 The Architecture

The DIMLP architecture illustrated in figure 1 differs from the standard MLP architecture in two main ways:

1. Each neuron in the first hidden layer is connected to only one input neuron.

2. The activation function used by the neurons of the first hidden layer is the staircase function instead of the sigmoid function.

---

1 The DIMLP software is available on ftp://ftp.comp.nus.edu.sg/pub/rstaff/guido.
As it will be explained in paragraph 2.3, staircase activation functions create linear frontiers. Moreover, with the restricted pattern of connectivity between the input layer and the first hidden layer, linear frontiers are parallel to the axis defined by input variables.

2.2 Learning

The training phase is carried out by varying the weights in order to minimize the \textit{Mean Squared Error} (MSE) function given as

\[ MSE = \frac{1}{2} \sum_p \sum_i (o_{pi} - t_{pi})^2; \]

where \( p \) represents the index of training examples, \( i \) is the index of output neurons and \( t_{pi} \) are target values for supervised learning. Although (1) is not differentiable with staircase activation functions we use a back-propagation algorithm. More precisely, during training the gradient is determined in each layer by the use of sigmoid functions, whereas the error related to the stopping criterion is calculated using staircase activation functions in the first hidden layer. Generally, because the staircase activation function represents a quantization of the sigmoid function, the difference of the responses given by a network with staircase activation functions and a network with sigmoid activation functions tends to zero when the number of stairs is sufficiently large. As an heuristic from experience, between 30 and 100 stairs are sufficient to learn a large number of data sets. The training algorithm is given as:

1. Forward an example using sigmoid activation functions in the first hidden layer.
2. Compute the gradient of the error.
3. Modify weights using the gradient.
4. If all examples have been presented compute the mean squared error of all examples using staircase activation functions in the first hidden layer.
5. If stop criterion reached (depending on the mean squared error) goto 1; else stop.

2.3 Rule Extraction

The key idea behind rule extraction is the precise localization of discriminant frontiers. In a standard multi-layer perceptron discriminant frontiers are not linear [3]; further, their precise localization is not straightforward. The use of staircase activation functions turns discriminant frontiers into well-determined hyper-planes. For clarity let us give two examples.

2.3.1 Examples

Figure 2 shows an example of DIMLP network with only one hidden layer and only one hidden neuron. Here, the staircase activation function is the step function given as

\[ h_1 = \begin{cases} 
1 & \text{if } w_0 + w_1x_1 \geq 0 \\
0 & \text{otherwise}
\end{cases} \]

(2)
The weight and the bias from the input layer to a hidden neuron define a \textit{virtual} hyper-plane frontier that will be effective (or also denoted to as \textit{real}) depending on the weights related to successive layers ($v_0$ and $v_1$ in figure 2). A real hyper-plane frontier will be located in $-w_0/w_1$. In fact, the points where $h_1 = 1$ are defined by $w_1 x_1 + w_0 \geq 0$. This is equivalent to $x_1 \geq -w_0/w_1$. The points where $h_1 = 0$ are defined by $w_1 x_1 + w_0 < 0$, which is equivalent to $x_1 < -w_0/w_1$.

Based on the network of figure 2, a real hyper-plane frontier is defined if

\[
\begin{cases}
  v_0 + v_1 \geq 0 & (h_1 = 1) \\
  v_0 < 0 & (h_1 = 0)
\end{cases}
\]  

Assuming that class \textit{black circle} is defined when $y_1 \geq 0.5$ and \textit{white square} otherwise, if (3) holds we obtain two symbolic rules given as

1. If \( x_1 \geq -w_0/w_1 \) \( \Rightarrow \) \textit{black circle}.

2. If \( x_1 < -w_0/w_1 \) \( \Rightarrow \) \textit{white square}.

Figure 3 illustrates another example. With the step activation function and with two hidden neurons there are two perpendicular hyper-planes. Note that for each rectangle the activations $h_1$ and $h_2$ are distinct logical values. To cover those logical expressions and to simplify them as much as possible we use a Karnaugh map. However, Karnaugh maps cannot be used for more than 6 logical variables. In next section a more general algorithm will be presented.

\subsection*{2.3.2 General Technique Description}

From previous examples it was shown that a DIMLP network builds a lattice of hyper-rectangles. Note that the number of hyper-rectangles is exponential with respect to the number of input attributes. Generally, the rule extraction task corresponds to the resolution of a covering problem where discriminant...
hyper-planes represent frontiers between regions of different classes. The purpose of the rule extraction algorithm is to cover with a minimal number of symbolic rules all hyper-rectangles containing training and testing examples. Even when we restrict the covering problem to available examples, the search for the minimal covering is NP-hard. However, sub-optimal solutions can be found in polynomial time by heuristic methods. Among several algorithms, it was decided to use decision trees as an important component of our rule extraction technique, because they are often faster than other algorithms.

Briefly, a decision tree is built by a recursive function splitting the input space by axis-parallel hyper-planes. At each step of the algorithm a criterion to determine the best split is used. This technique corresponds to a “Divide and Conquer” approach. One of the most popular decision tree models is C4.5 [18]. In a decision tree like C4.5 the inherent learning mechanism is based on an uni-variate search technique, whereas the training phase of a neural network is based on a multi-variate search. Therefore, during the training phase a decision tree may miss rules involving multiple attributes which are weakly predictive separately, but become strongly predictive in combination. On the other hand, a neural network may fail to discern a strongly relevant attribute among several irrelevant ones. Because we aim at using decision trees to extract rules from DIMLP networks the bias related to the decision tree search was modified by implementing the relevance hyper-plane criterion instead of the standard gain entropy measure [18].

The relevance of a discriminant hyper-plane corresponds to the number of points viewing this hyper-plane as the transition to a different class. In the first step of the covering algorithm the relevance of discriminant hyper-planes is estimated from all available examples and DIMLP responses. In practice, for all examples each input variable $x_i$ is varied by the quantity $\delta_i$, such as $\delta_i$ depends on the localization of next virtual hyper-plane.

Once the relevance of discriminant hyper-planes has been established a special decision tree is built according to the most relevant hyper-plane criterion. In other terms, during tree induction in a given region of the input space the hyper-plane having the largest number of points viewing this hyper-plane

---

**Figure 3.** An example of rule extraction with two input neurons. Here, the activation function of $h_1$ and $h_2$ is a step function. The partition built by the network into the bi-dimensional space is illustrated by A. In B, logical expressions represented by the activation values of the hidden neurons and the related classes are transcribed onto a Karnaugh map and then simplified. Finally, $h_1$ and $h_2$ are replaced by $x_1$ and $x_2$. 
as the transition to a different class is added to the tree.

Each path between the root and a leaf of the obtained decision tree corresponds to a rule. At this stage rules are disjointed and generally their number is large, as well as the number of antecedents. Therefore, a pruning strategy is applied to all rules according to the most enlarging pruned antecedent criterion. The use of this heuristic involves that at each step the pruning algorithm prunes the rule antecedent that most increases the number of carried examples without altering DIMLP classifications. Note that at the end of this stage, rules are no longer disjointed and unnecessary rules are removed.

When it is no longer possible to prune any antecedent or any rule, again, to increase the number of carried examples by each rule all thresholds of remaining antecedents are modified according to the most enlarging criterion. More precisely, for each attribute new threshold values are determined according to the list of discriminant hyper-planes. At each step, the new threshold of an antecedent that most increases the number of carried examples without altering DIMLP classifications is retained.

### 2.3.3 The Rule Extraction Algorithm

The rule extraction algorithm will generate rules which exactly represent neural network responses. The general algorithm is given as:

1. Determine virtual hyper-planes from weight values between the input layer and the first hidden layer.
2. Determine relevance of discriminant hyper-planes using available examples and DIMLP classifications.
3. Build a decision tree according to the most relevant hyper-plane criterion.
4. Prune rule antecedents according to the most enlarging pruned antecedent criterion.
5. Prune unnecessary rules.
6. Modify thresholds of antecedents according to the most enlarging criterion.

### 2.3.4 Algorithmic Complexity

Let us denote the number of inputs, $p$ the number of examples, $q$ the number of stairs in the staircase activation function, and $m$ the number of neurons in the first hidden layer divided by the number of inputs (we assume that the number of hidden neurons is a multiple of the number of inputs). The total number of virtual hyper-planes is $mnq$, therefore step 1 is $O(mnq)$. For step 2, each example is fed to the network and evaluated for each dimension along $mnq$ virtual hyper-planes in the worst case; therefore for all examples step 2 is $O(mnq)$. In the worst case of step 3, a tree with $p$ leaves, and with paths of length $mnq$ is built; again we have $O(mnq)$. For step 4 we have at most $p$ rules with $mnq$ antecedents. To prune all these antecedents the number of operations is $mnq + (mnq - 1) + ... + 1$, that is $O(mnq + (mnq)^2]$. In the worst case of step 5 there are $p$ rules; because each rule is removed sequentially this step is $O(p + p^2)$. Finally, for step 6 each threshold’s antecedent is replaced by $mq$ possible values in the worst case. Therefore, step 6 is $O(m^2n^2p^2q^4)$. It turns out that the computational complexity of the rule extraction algorithm is polynomial with respect to $m$, $n$, $p$, and $q$. 
Note that \( n \) and \( p \) are related to the size of the classification problem, whereas \( m \) is related to the size of the network.

3 Combinations of DIMLP Networks

Combining the predictions of several classifiers to produce a single classifier is generally more accurate than any of the individual classifier making up the ensemble. As shown in figure 4, our purpose is to use linear combinations of DIMLPs to improve the predictive accuracy of a single DIMLP network. More precisely, to build these ensembles we adopt the bagging and arcing methods introduced by Breiman [5, 6].

![Figure 4. A linear combination of DIMLP networks for a classification problem with two classes. In this particular case the response of each DIMLP network is averaged.](image)

3.1 Bagging and Arcing

Bagging and arcing are based on resampling techniques. Assuming that \( p \) is the size of the original training set, bagging generates for each classifier \( p \) examples drawn with replacement from the original training set. As a consequence, for each network many of the generated examples may be repeated while others may be left out. It is worth noting that the probability an an individual example from the training set will not be part of a re-sampled training set is \( (1 - 1/p)^p \approx 36.8\% \).

Arcing defines a probability to each example of the original training set. For each classifier the examples contained in the training set are selected according to these probabilities. More particularly, the probability \( q_i \) for selecting example \( i \) to be part of classifier \( K + 1 \)'s training set is defined as

\[
q_i = \frac{1 + m_i^4}{\sum_{j=1}^p (1 + m_j^4)}; \tag{4}
\]

where \( m_i \) is the number of times that example \( i \) is misclassified by the previous \( K \) classifiers.

3.2 Rule Extraction

Suppose we define a linear combination of two DIMLP networks and that from each network we extract only one rule. We denote the two rules to as \( R_1 \) and \( R_2 \). At the level of the overall combination...
(in the upper layer) the two rules are linearly combined. However, no intuitive meaning can be found in an expression given as $0.5 \cdot R_1 + 0.5 \cdot R_2$. This situation becomes even worse with many rules and many combined classifiers.

![Diagram](image)

**Figure 5.** An example of three DIMLP networks (D1, D2, and D3) combined with the “majority voting” paradigm. For each network the corresponding partition of the input space is shown in I, whereas II illustrates the overall voted partition. Note that each single network makes a mistake, but the ensemble of the three networks does not. Finally, III shows rule extraction performed by a Karnaugh map related to the hidden neurons of all networks.

The rule extraction technique presented in 2.3.3 can be applied to any DIMLP architecture having as many hidden layers as desired. In fact, the first hidden layer determines the exact location of hyper-planes, whereas all other layers just switch on or off discriminant frontiers in a given region of the input space. Since the overall combination of DIMLP networks is again a DIMLP network\(^2\) we can use the

\(^2\)For two consecutive layers and for two different networks of the ensemble all weights between the two networks are equal to zero.
same rule extraction technique. Here, all virtual hyper-planes defined by all DIMLP networks are taken
into account. The previous rule extraction technique (cf. 2.3.3) is modified in step 1. This step becomes:

1. Determine virtual hyper-planes from all DIMLP networks.

In this case the computational complexity of the algorithm is polynomial with respect to \( N, m, n, p, q \);
where \( N \) is the number of combined DIMLP networks, and \( m, n, p, q \) are the same constants defined in
paragraph 2.3.4. Again, fidelity of extracted rules is 100%.

4 Experiments

Four data sets were retrieved from the repository for Machine Learning at the University of California-
Irvine\(^3\). These are: 1. *Iris* [10]; 2. *Pima Indian Diabetes* [21]; 3. *Sonar* [13]; 4. *Vehicles* [9].

4.1 Methodology

Table 1 gives the characteristics of the data sets, as well as DIMLP neural architectures. The number
of neurons in the hidden layers is based on the heuristic that the number of weights is less than three times
the number of examples, and three neurons in the second hidden layer being a minimum. Ensembles with
25 DIMLP networks were defined and trained by bagging and arcing, as it was argued in [17] that the
improvement in accuracy is often achieved with the first 25 classifiers. Finally, DIMLP networks were
trained with default parameters. In our implementation of back-propagation default parameters are:
*learning parameter* = 0.1; *momentum* = 0.6; *flat spot elimination* = 0.01; *number of stairs* = 50.

<table>
<thead>
<tr>
<th>Table 1. Databases and DIMLP architectures.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cases</td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td>Diabetes</td>
</tr>
<tr>
<td>Iris</td>
</tr>
<tr>
<td>Sonar</td>
</tr>
<tr>
<td>Vehicles</td>
</tr>
</tbody>
</table>

Results are based on the average calculated after ten repetitions of ten-fold cross-validation for all data
sets. Further, the training phase of single DIMLP networks was stopped according to the minimal error
measured on an independent validation set. For each cross-validation trial the proportions of training
sets, validation sets, and testing sets were 8/10, 1/10 and 1/10, respectively. For each DIMLP network
of a bagged or an arced ensemble the validation set is defined by those examples that have not been
selected in the training set during the re-sampling stages (cf. 3.1). Thus, the proportions for the training
set and the validation set are 9/10, and again 1/10 of the examples are selected for the testing set. C4.5
decision trees with default parameters were also included in the experiments. Results for bagging and
arcing decision trees were those reported by Opitz and Maclin in [17]. Their results are based on five
repetitions of 10-fold cross-validation.

\(^3\)The internet site is ftp.ics.uci.edu.
4.2 Results

Predictive accuracies are given in table 2. For each classification problem we computed the standard error of the difference between the best average accuracy of neural networks and the best one of decision trees. The $t$ statistic for testing the null hypothesis that the two means are equal was then obtained and a two-tailed test was conducted. If the null hypothesis was rejected at the level of 1%, we emphasized in table 2 the method that gave the highest value. DIMLP ensembles built by bagging were significantly more accurate on average in the first problem, whereas in the last two problems arced DIMLPs obtained the best results.

<table>
<thead>
<tr>
<th>Table 2. Predictive accuracies.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIMLP</td>
</tr>
<tr>
<td>Diabetes</td>
</tr>
<tr>
<td>Iris</td>
</tr>
<tr>
<td>Sonar</td>
</tr>
<tr>
<td>Vehicles</td>
</tr>
</tbody>
</table>

We estimated the complexity of a classifier by the average number of extracted antecedents per rule set. Results are presented in table 3. To the best of our knowledge it is not yet possible to use a reliable algorithm to extract symbolic rules from a large number of combined decision trees [19]. Rules extracted from arced DIMLPs were significantly more accurate than those extracted from C4.5 decision trees in the Sonar problem (84.2% versus 71.9%) and in the Vehicle problem (79.7% versus 72.4%). Nevertheless, rules extracted from C4.5 exhibited lower complexity. Note that for the Iris problem and the Sonar problem the number of extracted antecedents per rule set from bagged DIMLPs was almost similar to those generated from decision trees. Figure 6 presents an example of rule set extracted from bagged DIMLPs applied to the Iris problem.

<table>
<thead>
<tr>
<th>Table 3. Complexity of extracted rules in terms of number of antecedents per rule set.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIMLP</td>
</tr>
<tr>
<td>Diabetes</td>
</tr>
<tr>
<td>Iris</td>
</tr>
<tr>
<td>Sonar</td>
</tr>
<tr>
<td>Vehicles</td>
</tr>
</tbody>
</table>

4.3 Discussion of Results

With respect to a single DIMLP network, bagging has built ensemble classifiers with lower complexity (Diabetes problem), similar complexity (Iris and Sonar problems), and slightly greater complexity (Vehicles problem). At least for these 4 classification problems bagging can thus be seen as a mechanism to reduce, maintain, or slightly increase single classifier complexity. By contrast, arcing significantly increased the complexity of DIMLP ensembles. It is worth noting that classifiers with the highest complexity resulted the most accurate for the Sonar and Vehicle classification problems. This is surprising
| Rule 1: If ($x_3 < -0.443$) $\implies$ Class 3 (50) |
| Rule 2: If ($x_3 > -0.443$) and ($x_3 < 0.700$) and ($x_4 < 0.659$) $\implies$ Class 2 (47) |
| Rule 3: If ($x_3 > 0.700$) $\implies$ Class 1 (46) |
| Rule 4: If ($x_2 < 0.337$) and ($x_4 > 0.527$) $\implies$ Class 1 (36) |
| Rule 5: If ($x_2 > 0.104$) and ($x_3 > 0.362$) and ($x_3 < 0.700$) $\implies$ Class 2 (8) |

| Default Rule: Class 1 (16) |

**Figure 6.** Two rule sets generated from an ensemble of bagged DIMLP networks trained on the Iris problem. The rule extraction algorithm first generated a set of unordered rules (“if-then” rules) and then a set of ordered rules (“if-then-else” rules). Accuracies were 98.5% on the training set, and 100% on the testing set for both rule sets.

with respect to the Occam’s razor argument for which the better classifier is the simplest one. Our results support Webb’s arguments against Occam’s Razor [24], as well as Quinlan’s experiments in [19].

## 5 Conclusion

We presented a rule extraction technique to explain DIMLP responses with 100% fidelity. The same technique was used to extract symbolic rules from ensembles of DIMLP networks. Generated rules were significantly more accurate than those extracted from C4.5 decision trees on three problems. To our knowledge, ensembles of decision trees do not present any satisfactory explanation capability, whereas DIMLP networks are transparent when they are also combined in an ensemble.

### References


