

# Analysis of hidden representations by greedy clustering

Rudy Setiono and Huan Liu

Department of Information Systems and Computer Science

National University of Singapore

Kent Ridge, Singapore 119260

Republic of Singapore

Email:{rudys,liuh}@iscs.nus.edu.sg

## Abstract

The hidden layer of backpropagation neural networks holds the key to the networks' success in solving pattern classification problems. The units in the hidden layer encapsulate the network's internal representations of the outside world described by the input data. In this paper, the hidden representations of trained networks are investigated by means of a simple greedy clustering algorithm. This clustering algorithm is applied to networks that have been trained to solve well known problems: the monks problems, the 5-bit parity problem and the contiguity problem. The results from applying the algorithm on problems with known concepts provide us with a better understanding of neural network learning. These results also explain why neural networks achieve higher predictive accuracy than that of decision tree methods. The results of this study can be readily applied to rule extraction from neural networks. Production rules are extracted for the parity and the monks problems, as well as for a benchmark data set: Pima Indian diabetes diagnosis. The extracted rules from the Pima Indian diabetes data set compare favorably with rules extracted from ARTMAP neural networks in terms of predictive accuracy and simplicity.

*Keywords:* Backpropagation neural network, hidden representation, pruning, clustering, rule extraction.

## 1 Introduction

The problem of distinguishing patterns from two or more disjoint sets is one of the areas where backpropagation neural networks have been extensively applied. For this pattern classification problem, neural networks have been shown by many researchers to be a viable alternative to the traditional decision tree methods (Dietterich *et al.*, 1990; Fisher & McKusick, 1989; Quinlan, 1994; Shavlik *et al.*, 1990; Weiss & Kapouleas, 1992). Although the neural network approach requires more computation time than the decision tree methods, it often provides higher accuracy rates (Quinlan,

1994). One drawback of the neural network approach is that the decision process of a neural network, unlike that of a decision tree, is hard to interpret.

In recent years, there have been a large number of published works focusing on algorithms that extract rules from trained neural networks (Andrews *et al.*, 1995). The reason behind the surge of interest among researchers in neural network and machine learning community is clear. It is no longer satisfactory to obtain only high accuracy from a network; it is also desirable to be able to explain the decision process of the network. This decision process is normally explained by a set of rules that is extracted from the network. A set of extracted rules can be verified by a human expert and it may also provide novel insights into the problem.

The internal representations of the data visible to the input and output layers are embedded in the units of the hidden layer of a trained network. The fundamental idea behind many algorithms that extract rules from trained networks (Fu, 1991; Towell & Shavlik, 1993; Blassig, 1994; Setiono & Liu, 1996) is to analyze the hidden unit activation values of the network. In this paper, we present our effort in explaining why neural networks can have better performance than decision tree methods by examining these hidden representations more closely. We apply a greedy clustering algorithm to the hidden units of networks that have been trained to solve several synthetic problems: the monks problems (Thrun *et al.*, 1991), the 5-bit parity problem and the contiguity problem (Denker *et al.*, 1987). We choose these problems, where the concepts to be learned are known, to better understand how neural networks group the input patterns for classification. The intermediate concepts of the problem domain learned by the units in the hidden layer can be revealed by relating each cluster formed by the clustering algorithm to its relevant input units. We also apply the algorithm to a benchmark problem: the Pima Indian diabetes diagnosis.

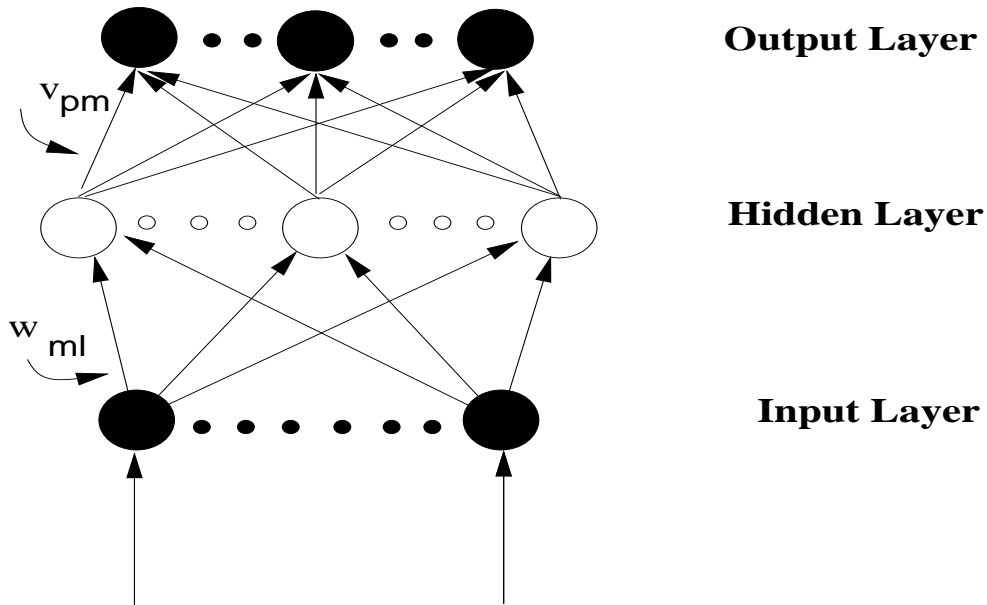
The results of our study indicate that the robustness of neural networks learning can be attributed to the following reasons:

1. the contribution of all relevant input attributes to classification being considered together as a group;
2. the use of all available input patterns in each epoch of the training process;
3. the vertical and/or horizontal data reduction<sup>1</sup>.

The organization of the paper is as follows. We describe our neural networks training and pruning algorithms in Section 2. The greedy clustering algorithm for

---

<sup>1</sup>Vertical reduction is achieved when the number of unique hidden representations is less than the number of original patterns in the data set, while horizontal reduction is achieved when the pruned network has fewer hidden units than input units.



**Figure 1.** Fully connected feedforward neural network with 3 hidden units.

analyzing the hidden representations of a trained network is presented in Section 3. We present the results from applying the clustering algorithm on the monks problems, the 5-bit parity problem and the contiguity problem in Section 4. Experimental results using the Pima Indian diabetes diagnosis data set are described in Section 5. A discussion of our results and a comparison between neural networks with the decision tree methods are given in Section 6. Finally, Section 7 concludes the paper.

## 2 Neural network training and pruning

The analysis described in this paper considers only the commonly used network architecture, namely, the three-layered feedforward networks. Figure 1 depicts one such network.

Given the training input-output pairs, the training of the neural network entails finding a set of weights  $(w, v)$  that minimizes the network error function. The error function that we minimize is the cross-entropy function:

$$F(w, v) = - \sum_{i=1}^k \sum_{p=1}^C t_{pi} \log S_{pi} + (1 - t_{pi}) \log(1 - S_{pi}) \quad (1)$$

where

- $k$  is the number of patterns.
- $C$  is the number of output units.

- $t_{pi} = 0$  or  $1$  is the target value for pattern  $x_i$  at output unit  $p$ ,  $p = 1, 2, \dots, C$ .
- $S_{pi}$  is the output of the network at output unit  $p$ .

$$S_{pi} = \sigma \left( \sum_{m=1}^H \theta \left( (x_i)^T w_m \right) v_{pm} \right) \quad (2)$$

- $\theta(\xi)$  is the hidden unit activation function, it is either the sigmoid function  $\sigma(\xi) = 1/(1 + e^{-\xi})$  or the hyperbolic tangent function  $\psi(\xi) = (e^\xi - e^{-\xi})/(e^\xi + e^{-\xi})$ .
- $H$  is the number of hidden units in the network.
- $x_i$  is an  $n$ -dimensional input pattern,  $i = 1, 2, \dots, k$ .
- $w_m$  is an  $n$ -dimensional vector of weights for the arcs connecting the input layer and the  $m$ -th hidden unit,  $m = 1, 2, \dots, h$ . The weight of the connection from the  $\ell$ -th input unit to the  $m$ -th hidden unit is denoted by  $w_{m\ell}$ .
- $v_m$  is a  $C$ -dimensional vector of weights for the arcs connecting the  $m$ -th hidden unit and the output layer. The weight of the connection from the  $m$ -th hidden unit to the  $p$ -th output unit is denoted by  $v_{pm}$ .

Experimental results have indicated that this cross-entropy error function enjoys higher convergence than the normal sum of squared error function (Lang & Witbrock, 1989; van Ooyen & Nienhuis, 1992).

In order to eliminate irrelevant input and hidden units, a trained network needs to be trimmed by pruning. The pruning algorithm must be able to find those connections in the network that are redundant. Removing the redundant connections from the network usually increases the capability of the network to generalize, i.e., to predict new patterns not used for training with a higher accuracy rate. An input unit that has all its connections removed can be eliminated without affecting the network accuracy. Similarly, a hidden unit that has all its output connections eliminated should also be removed from the network.

A weight-decay term is normally added to the error function so that connections that are redundant will have weights with small magnitude at the end of training. The weight-decay term that we use is

$$P(w, v) = \epsilon_1 \sum_{m=1}^H \left( \sum_{\ell=1}^n \frac{\beta w_{m\ell}^2}{1 + \beta w_{m\ell}^2} + \sum_{p=1}^C \frac{\beta v_{pm}^2}{1 + \beta v_{pm}^2} \right) + \epsilon_2 \sum_{m=1}^H \left( \sum_{\ell=1}^n w_{m\ell}^2 + \sum_{p=1}^C v_{pm}^2 \right) \quad (3)$$

( $\epsilon_1, \epsilon_2$  and  $\beta$  are positive parameters). Network connections are removed based on their magnitude. The details of the pruning algorithm that makes use of the weight-decay term (3) can be found in our earlier work (Setiono, 1997). The pruning algorithm is shown to be very successful in removing redundant and irrelevant connections for the many problems tested.

### 3 A greedy clustering algorithm

Let us assume that the number of hidden units in the pruned network is  $H$ . Clustering the hidden unit activation values is accomplished by a simple greedy algorithm which can be summarized as follows.

---

#### A greedy clustering algorithm (GCA)

1. Find the smallest positive integer  $d$  such that if all the network activation values are rounded to  $d$ -decimal-place, the network still retains its accuracy rate.
2. Represent each activation value  $\alpha$  by the integer closest to  $\alpha \times 10^d$ . Let  $\mathcal{H}_i = \langle h_{i,1}, h_{i,2}, \dots, h_{i,k} \rangle$  be the  $k$ -dimensional vector of these representations at hidden unit  $i$  for patterns  $x_1, x_2, \dots, x_k$  and let  $\mathcal{H} = [\mathcal{H}_1, \mathcal{H}_2, \dots, \mathcal{H}_H]$  be the  $k \times H$  matrix of the hidden representations of all patterns at all  $H$  hidden units.
3. Let  $P$  be a permutation of the set  $\{1, 2, \dots, H\}$  and set  $m = 1$ .
4. Set  $i = P(m)$ .
5. Sort the values of the  $i$ -th column ( $\mathcal{H}_i$ ) of matrix  $\mathcal{H}$  in increasing order.
6. Find a pair of distinct adjacent values  $h_{i,j}$  and  $h_{i,j+1}$  in  $\mathcal{H}_i$  such that if  $h_{i,j+1}$  is replaced by  $h_{i,j}$ , no conflicting data will be generated <sup>2</sup>.
7. If such a pair of values exists, replace all occurrences of  $h_{i,j+1}$  in  $\mathcal{H}_i$  by  $h_{i,j}$  and repeat Step 6. Otherwise, set  $m = m + 1$ . If  $m \leq H$ , go to Step 4, else Stop.

---

If the activation value of an input pattern at hidden unit  $m$  is computed as the sigmoid of the weighted input  $\sigma(\sum_{\ell=1}^n x_{i\ell} w_{m\ell})$ , it will have a value in the interval  $[0, 1]$ . If the hyperbolic tangent function is used instead, then the range of activation values is the interval  $[-1, 1]$ . Steps 1 and 2 of GCA find integer representations of all

---

<sup>2</sup>Conflicting data occur when there are two or more identical hidden representations for patterns that belong to different classes.

hidden unit activation values. A small value for  $d$  in Step 1 indicates that relatively few distinct values for the activation values are sufficient for the network to maintain its accuracy. For example, when  $d = 2$ , the distinct values are  $0.00, 0.01, 0.02, \dots, 1.00$  if the sigmoid function is used. In general, there could be up to  $10^d + 1$  distinct values. If the hyperbolic tangent function is used and  $d$  is 2, then there could be up to 201 distinct values:  $-1.00, -0.99, -0.98, \dots, 0.99, 1.00$ . For the results reported in this paper, we set the value of  $d$  to 2.

The array  $P$  contains the sequence in which the hidden units of the network are to be considered. Different ordering sequences usually result in different clusters of activation values. Once a hidden unit is selected for clustering, the discretized activation values are sorted in Step 5 such that the activation values are in increasing order. The values are clustered based on their distance. We implemented Step 6 of the algorithm by first finding a pair of adjacent distinct values with the shortest distance. If these two values can be merged without introducing conflicting data, they will be merged. Otherwise, a pair with the second shortest distance will be considered. This process is repeated until there is no more pair of values that can be merged. The next hidden unit as determined by the array  $P$  will then be considered.

## 4 Empirical study

Three artificial problems have often been used by researchers in neural networks and machine learning. These problems are the monks problems (Thrun *et al.*, 1991), the  $n$ -bit parity problems and the contiguity problem (Denker *et al.*, 1987). The monks problems are artificial robot domains, in which robots are described by six different attributes:

$\mathcal{A}_1$ : head_shape $\in$ round, square, octagon;	$\mathcal{A}_2$ : body_shape $\in$ round, square, octagon;
$\mathcal{A}_3$ : is_smiling $\in$ yes, no;	$\mathcal{A}_4$ : holding $\in$ sword, balloon, flag;
$\mathcal{A}_5$ : jacket_color $\in$ red, yellow, green, blue;	$\mathcal{A}_6$ : has_tie $\in$ yes, no.

There are 3 monks problems and their learning tasks are of binary classification, each of them is given by the following logical description of a class.

- Problem Monks 1: (head\_shape = body\_shape) or (jacket\_color = red). From 432 possible patterns, 124 were randomly selected for the training set.
- Problem Monks 2: Exactly two of the six attributes have their first value. From 432 patterns, 169 were selected randomly for the training set.

- Problem Monks 3: (Jacket\_color is green and holding a sword) or (jacket\_color is not blue and body\_shape is not octagon). From 432 patterns, 122 were selected randomly for training and among them there were 6 misclassifications, 5 monks were labeled as non-monks, and 1 non-monk was labeled as monk.

The inputs are coded in binary. Each attribute with  $n$  possible values are coded as  $n$ -bit binary string with exactly 1 bit equal to 1 and the rest equal to 0. For example, head\_shape = round is represented by (0,0,1), while head\_shape = square by (0,1,0).

The input set of the  $n$ -bit parity problem consists of  $2^n$  patterns in the  $n$ -dimensional space and each pattern is an  $n$ -bit binary string. The target value  $t_i$  is equal to 1 if the number of one's in the pattern is odd and it is 0 otherwise.

The data set of the contiguity problem also consists of binary strings with length  $n$ . Patterns are labeled based on the number of clumps that they have. A clump is a consecutive sequence of +1's. The patterns are divided into two classes: those with two clumps and those with three clumps. The target value is 1 if a pattern contains 3 clumps, and it is 0 if the pattern contains 2 clumps.

All networks were trained using the BFGS optimization algorithm (Setiono, 1995). Initial weights of the network connections were generated randomly in the interval  $[-1, 1]$ . The values of the parameters  $\epsilon_1, \epsilon_2$  and  $\beta$  in the weight-decay term (3) were set to 0.1,  $10^{-4}$  and 10, respectively.

For the monks and parity problems, the hidden unit activation function used was the sigmoid function. For the contiguity problem and the Pima Indian diabetes problem, the hyperbolic tangent function was used as the hidden unit activation function. We begin with the analysis of neural networks that have been trained to solve the Monks 2 problem. For this problem, the neural network approach was shown to perform significantly better than 20 other classification methods (Thrun *et al.*, 1991).

## 4.1 The Monks 2 problem

We trained 100 neural networks to solve the problem. Each of the network had 1 output unit, 4 hidden units, and 18 input units. The 18-th input was used for the bias or threshold of the hidden units, its input value for all patterns was set to 1. Each network was trained to achieve 100% accuracy rate on the training patterns. The redundant connections and units of the network were then removed by pruning. A connection was removed only if the resulting network could still predict all the training patterns correctly. Hence, when the pruning terminated, the accuracy of the network was still 100%. The results are summarized in Table I.

**Table I.** Statistics of 100 pruned networks for the Monks 2 problem. Figures in parentheses denote standard deviations. The accuracy of all networks on the training data before and after pruning is 100 %.

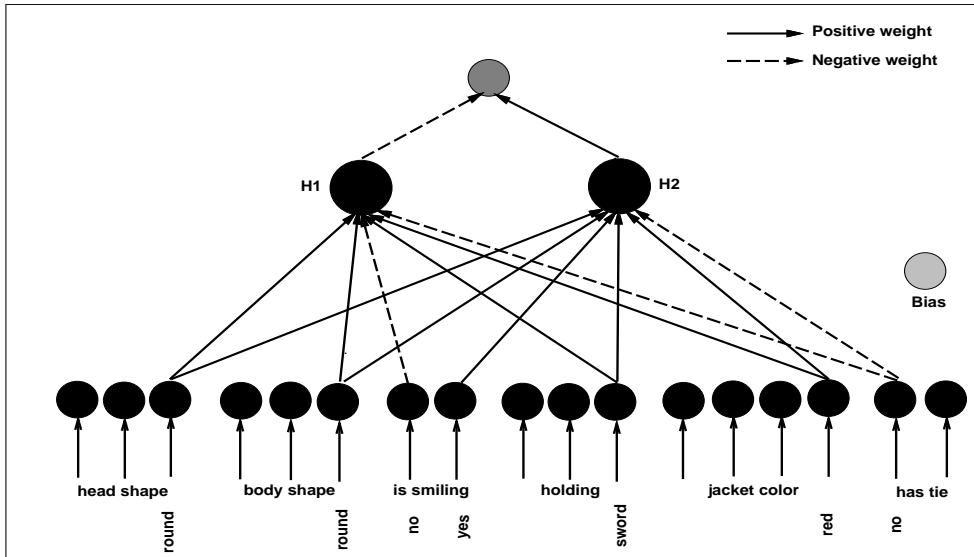
<b>Monks 2</b>	
<b>Before pruning</b>	
Number of connections	76
Number of hidden units	4
Ave. accuracy on testing set (%)	98.87 (1.33)
<b>After pruning</b>	
Average no. of connections	15.63 (1.15)
Average no. of hidden units	2.61 (0.49)
Ave. accuracy on testing set (%)	99.64 (0.73)

The hidden unit activation values of all networks were then clustered by GCA. All possible orderings of the hidden units were considered. Hence, for instance, if there were 3 hidden units left in the network after pruning, we run GCA 6 ( $= 3!$ ) times with different ordering sequence P (Step 3). Three pieces of information were collected from each run: (1) the number of hidden units whose activation values are *not* clustered into a single value, (2) the number of possible combinations of clusters, and (3) the number of unique hidden representations of the training patterns. If the number of hidden units is  $H$  and the number of clusters in the hidden units are  $C_1, C_2, \dots, C_H$ , then the maximum number of combinations is  $C_1 \times C_2 \dots \times C_H$ . However, some of the possible combinations may not occur in the data set.

Thirty-nine of the pruned networks had 2 hidden units left and the remaining sixty-one had 3 hidden units. Hence, GCA was run  $39 \times 2! + 61 \times 3! = 444$  times. The results are summarized in Table II.

**Table II.** Results of 444 runs of GCA on hidden representations of 100 neural networks for Monks 2 problem. The numbers of hidden units denote the hidden units whose activation values are *not* clustered into a single value. The numbers in parentheses indicate the frequency (out of 444 runs).

No. of hidden units	1(231), 2(189), 3(24)
No. of possible cluster combinations	3(227), 4(162), 5(4), 6(8), 8(22), 9(2), 12(19)
No. of unique representations	3(386), 4(3), 5(17), 6(3), 7(19), 8(16)



**Figure 2.** A pruned network that solves the Monks 2 problem.

The figures in Table II reveal the following:

1. The number of hidden units can be further reduced after clustering. The hidden representations of a very few hidden units provide sufficient information to distinguish between monks and non-monks. In fact, for the majority of the runs (231), the information contained in 1 hidden unit alone is sufficient.
2. The 169 training patterns from a relatively complex problem domain can be reduced to as few as 3 unique one-dimensional hidden representations by a neural network.

For each of the 100 networks, we also recorded the smallest number of hidden representations obtained from one of its  $H!$  GCA runs. The hidden units of 92 of the 100 networks could reduce the training patterns to just 3 unique representations, those of 5 networks to 7 representations. The remaining 3 networks reduced the data to 4, 5, and 6 representations, respectively.

In order to investigate further how a network solves the problem, we select a typical pruned network and analyze its hidden representations more closely. This network is depicted in Figure 2.

Since there are 2 hidden units in the network, we run GCA twice. In the first run, we considered the activation values of the first (left) hidden unit, then those of the second (right) hidden unit. In the second run, the order was reversed.

1. H1, then H2.

GCA found that two distinct values at each of the two hidden units were suf-

ficient to separate the monks from the non-monks. The values at H1 were 23 and 64, while the values at H2 were 0 and 100. Hence, H1 divided the patterns into two sets, those with activation values in the interval  $[0.23, 0.635)$  and those in the interval  $[0.635, 1]$ . All patterns with their H1 activation values in  $[0.23, 0.635)$  were assigned a hidden representation value of 23, while those in the interval  $[0.635, 1]$  were given a value of 64. Similarly, H2 divided the patterns into two groups, those with activation values in the interval  $[0.0, 0.995)$  and those with activation values in the interval  $[0.995, 1]$ . Hence, each H2 activation value was represented by either 0 or 100. The distribution of the training patterns by the hidden units are summarized in Table III

**Table III.** Results from GCA: H1 is clustered first, followed by H2. M = monk, N = not monk.

<b>H1</b>		<b>H2</b>	
Interval 1	Interval 2	Interval 1	Interval 2
$[0.23, 0.635)$	$[0.635, 1]$	$[0.0, 0.995)$	$[0.995, 1.0]$
41 N/64 M	64 N	41 N	64 N/64 M

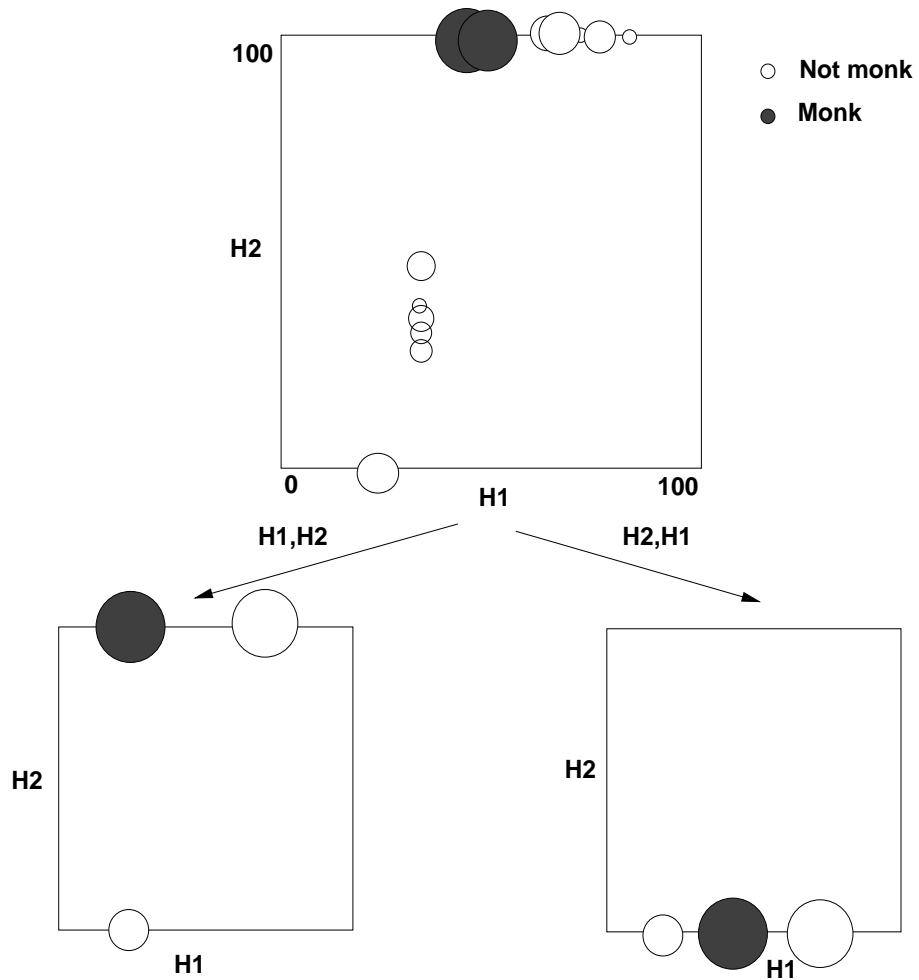
2. H2, then H1.

GCA found that the activation values in the first hidden unit could be replaced by one of the three values: 23, 49, or 64, while all values in the second hidden unit could be replaced by just a single value, 0. Hence, the hidden representations of the first hidden unit alone are sufficient to separate the monks from the non-monks. The distribution of the training patterns by the first hidden unit is summarized in Table IV.

**Table IV.** Results from GCA: H2 is clustered first, followed by H1.

<b>H1</b>			<b>H2</b>
Interval 1	Interval 2	Interval 3	Interval 1
$[0.23, 0.485)$	$[0.485, 0.635)$	$[0.635, 1.0]$	$[0.0, 1.0]$
41 N	64 M	64 N	105 N/64 M

Figure 3 shows the scatter diagrams of the network hidden unit activation values before and after clustering. The top diagram clearly demonstrates the robustness



**Figure 3.** Hidden representations of the Monks 2 data. Top diagram shows the activation values of the training patterns. Bottom diagrams show the activation values after clustering using 2 different ordering sequences. Sizes of the circles denote relative counts.

of the neural network approach. A set of 169 training patterns, each of which has 17 binary attributes, is reduced to just a few patterns in the 2-dimensional space. The network achieves this horizontal and vertical data reduction by discovering the relevant input attributes and a set of weights for the network connections between the inputs and the hidden layer. The two lower diagrams show how the hidden representations could be made even more compact by clustering. Both diagrams lead us to conclude that the original concept in the data can be captured in as few as 3 hidden representations. It needs be to emphasized that the hidden unit activation values of hidden unit 1 alone contain all the information required for classification. However, this does not imply that a network with just 1 hidden unit can solve the problem. Clustering is done in order to analyze how the inputs of monks and of non-monks are mapped by the network weights from the input layer to the hidden units. We do not make use of the weights of the network from the hidden layer to the output layer during clustering. The result from using the “no conflicting data” criterion in Step 6 of GCA is that, in many networks, the activation values of all hidden units but one are clustered to a single value.

So few clusters lead us naturally to rule extraction. By making use of the network hidden representations, rules can be generated to explain the network outputs in terms of its inputs in two stages. To illustrate this point, we select the hidden representations depicted in the lower right corner of Figure 3. It is evident from the diagram that monks and non-monks are distinguished solely based on their first hidden unit activation values. A pattern is classified as a monk only if its activation values falls the interval  $[0.485, 0.635)$  (Table IV). The rules are then:

Rule 1. If the activation value at H1 is in  $[0.485, 0.635)$ , then **monk**.

Default rule: **not monk**.

The activation values of the patterns are determined by the weights of the relevant inputs to hidden unit H1. An activation value will fall in the interval  $[0.485, 0.635)$  if and only if

$$0.485 \leq \sigma(0.6\mathcal{I}_3 + 0.6\mathcal{I}_6 - 0.6\mathcal{I}_7 + 0.6\mathcal{I}_{11} + 0.6\mathcal{I}_{15} - 0.6\mathcal{I}_{16}) < 0.635$$

Since  $\sigma(-0.06) = 0.485$  and  $\sigma(0.55) = 0.635$ , the above inequalities are equivalent to

$$-0.06 \leq 0.6\mathcal{I}_3 + 0.6\mathcal{I}_6 - 0.6\mathcal{I}_7 + 0.6\mathcal{I}_{11} + 0.6\mathcal{I}_{15} - 0.6\mathcal{I}_{16} < 0.55 \quad (4)$$

(The binary input  $\mathcal{I}_3$  is 1 iff head\_shape is round,  $\mathcal{I}_6$  is 1 iff body\_shape is round,  $\mathcal{I}_7$  is 1 iff is\_smiling is no,  $\mathcal{I}_{11}$  is 1 iff holding is sword,  $\mathcal{I}_{15}$  is 1 iff jacket\_color is red, and  $\mathcal{I}_{16}$  is 1 iff has\_tie is no.) Let us divide the relevant inputs into two sets, those with

positive coefficients  $\mathcal{I}_+ = \{\mathcal{I}_3, \mathcal{I}_6, \mathcal{I}_{11}, \mathcal{I}_{15}\}$  and those with negative coefficients  $\mathcal{I}_- = \{\mathcal{I}_7, \mathcal{I}_{16}\}$ . Similarly, let us split the original set of six attributes into two sets,  $\mathcal{A}_+ = \{\text{head\_shape}, \text{body\_shape}, \text{holding}, \text{jacket\_color}\}$  and  $\mathcal{A}_- = \{\text{is\_smiling}, \text{has\_tie}\}$ . Note that an input in  $I_+$  equals to 1 implies the corresponding original attribute in  $A_+$  has its first value and an input in  $I_-$  equals to 1 implies the corresponding attribute in the set  $A_-$  has its second value. It is apparent from condition (4) that a pattern is classified as a monk only if not more than 2 of its inputs in the set  $\mathcal{I}_+$  have values of 1. We consider the 3 cases where a pattern can possibly be classified as a monk:

1. Two inputs in the set  $\mathcal{I}_+$  have values equal to 1. In this case, both inputs in the set  $\mathcal{I}_-$  must also be 1 in order to satisfy the inequalities (4). Hence, if two attributes in the set  $\mathcal{A}_+$  have their first values, both attributes in the set  $\mathcal{A}_-$  must have their second values.
2. Exactly one input in the set  $\mathcal{I}_+$  has value equal to 1. In this case, exactly one of the two inputs in the set  $\mathcal{I}_-$  must also be 1. It follows that if one attribute in the set  $\mathcal{A}_+$  has its first value, then exactly one of the two attributes in the set  $\mathcal{A}_-$  must also have its first value.
3. None of the four inputs in the set  $\mathcal{I}_+$  has value equal to 1. To satisfy equation 4, no input in the set  $\mathcal{I}_-$  can have value equal to 1 either. In this case, both attributes in the set  $\mathcal{A}_-$  have their first values, and none of the attributes in the set  $\mathcal{A}_+$  has their first values.

Hence, from the analysis of the hidden representations of the pruned network, it is possible to explain the network decision process in two ways. First, as a set of rules involving the weights of the network that define the decision boundaries:

Rule 1. If  $-0.06 \leq 0.6\mathcal{I}_3 + 0.6\mathcal{I}_6 - 0.6\mathcal{I}_7 + 0.6\mathcal{I}_{11} + 0.6\mathcal{I}_{15} - 0.6\mathcal{I}_{16} < 0.55$ , then **monk**.

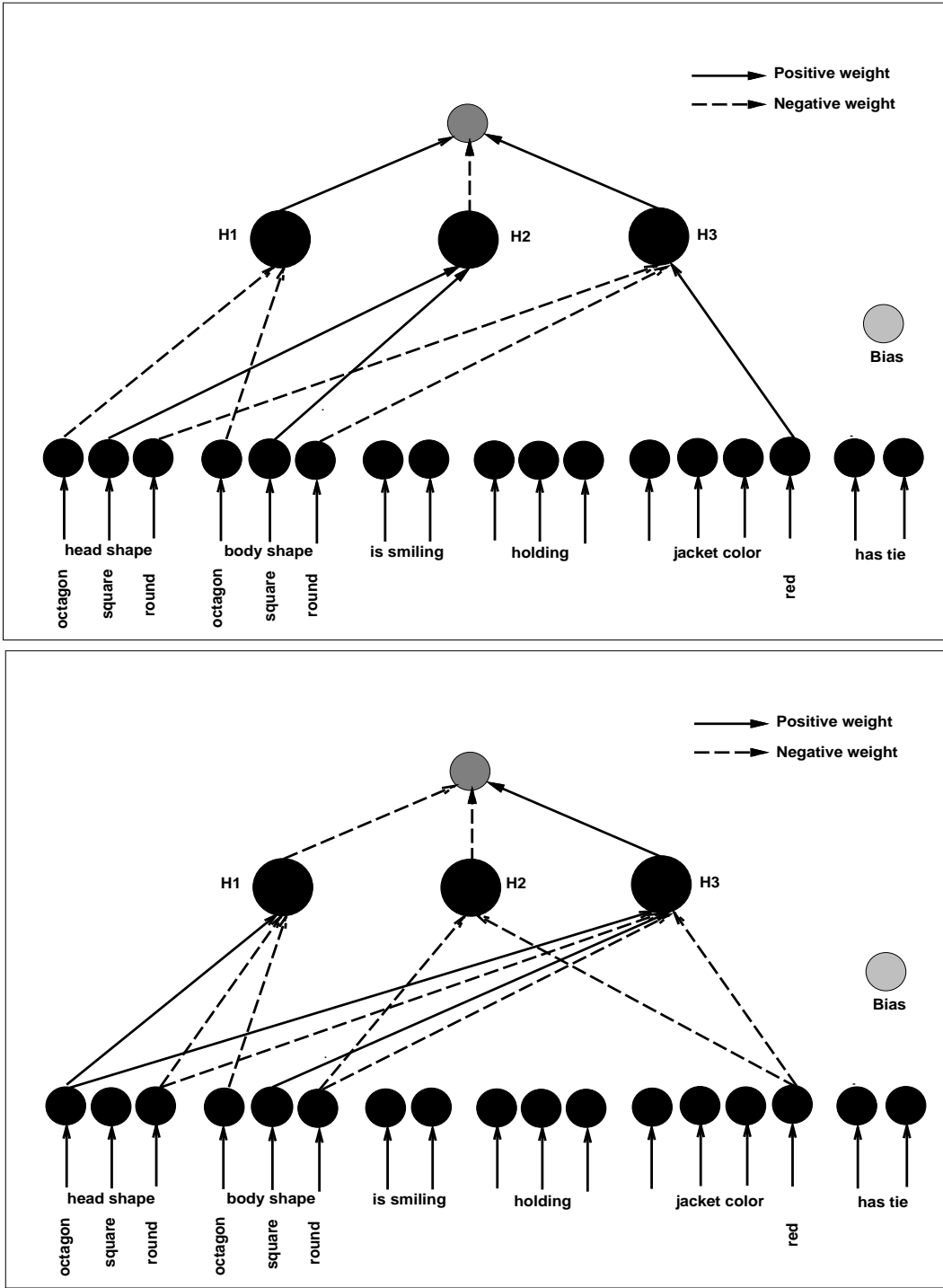
Default rule: **not monk**.

Second, as a set of symbolic rules that does not involve any network weights to describe the possible 3 cases that follow from the first set of rules:

Rule 1. If exactly two of the six attributes have their first value, then **monk**.

Default rule: **not monk**.

For the purpose of comparison, we also solved the problem using C4.5 (Quinlan, 1993). C4.5 is a widely-used program that generates rules from decision trees. The total number of rules generated was 10 and the number of conditions in each rule was either 2,3, or 4. The rules achieved 82.8 % accuracy rate on the 169 training patterns and only 66.7 % on the testing patterns.



**Figure 4.** Two networks that solve the Monks 1 problem. Network 1 (top) has the fewest connections among the 100 pruned networks, while Network 2 (bottom) has the activation values in two of its three hidden units clustered into a single value.

Table V. Results of GCA for the two networks in Figure 4.

Network 1						
H1		H2		H3		
Interval 1	Interval 2	Interval 1	Interval 2	Interval 1	Interval 2	Interval 3
[0.0, 0.495)	[0.495, 1.00]	[0.495, 1.0)	[1.0, 1.0]	[0.0, 0.495)	[0.495, 0.895)	[0.895, 1.0]
33 N/34 M	29 N/28 M	22 N/33 M	40 N/29 M	51 N/9 M	11 N/25 M	28 M
Network 2						
H1	H2	H3				
Interval 1	Interval 1	Interval 1	Interval 2	Interval 3	Interval 4	Interval 5
[0.0, 1.0]	[0.0, 1.0]	[0.04, 0.145)	[0.145, 0.605)	[0.605, 0.865)	[0.865, 0.905)	[0.905, 1.0]
62 N/62 M	62 N/62 M	8 M	57 N	26 M	5 N	28 M

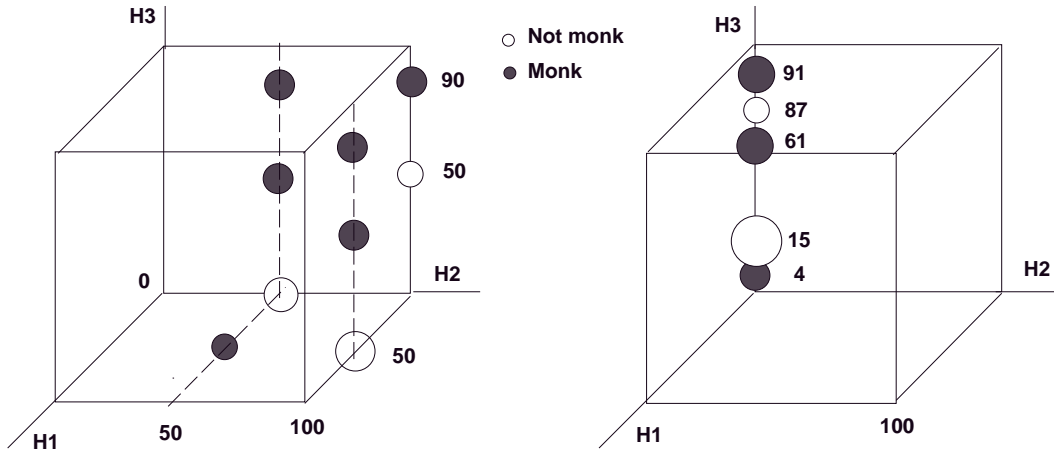


Figure 5. Hidden representations of the Monks 1 data by two networks shown in Figure 4. Network 1 represented each training pattern by one of the nine three-dimensional clusters. Network 2 represented a training pattern by one of the 5 one-dimensional clusters.

## 4.2 The Monks 1 problem

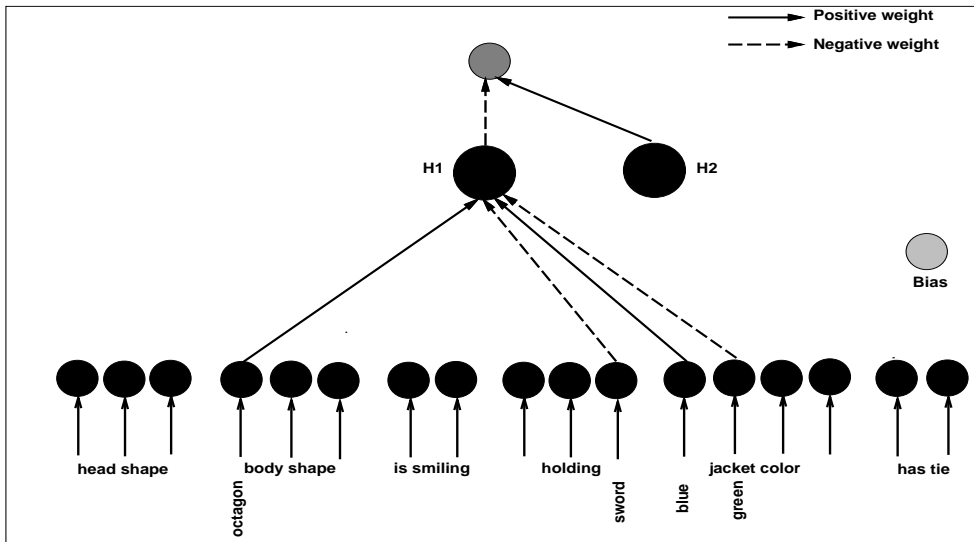
We also trained 100 networks for this problem. The topology of each network was the same as that for the Monks 2 problem described in the previous subsection. After pruning the average number of connections left was 13.14, while the average number of hidden units was 3.42. Compared with the Monks 2 problem, the average number of hidden units is higher. Running GCA for all possible ordering sequence of the hidden units produced more variation in the resulting hidden representations. The number of unique hidden representations of the training patterns in the hidden layer ranged from 3 to 20.

A network with the fewest connections is shown in Figure 4, Network 1(top). Network 2 (bottom) is one of the networks whose activation values in all but one hidden unit can be represented by a single value (cf. Table V). Both networks achieve 100% accuracy rates on the training and testing data sets. Table V summarizes the results of one run of GCA on the two networks. In Network 1, the numbers of representations at the three hidden units were 2, 2 and 3 respectively. Of the twelve possible combinations of these hidden unit representations, 9 were present in the training data set. In Network 2, the number discrete representations was 1 at hidden unit 1 and 2, and it was 5 at hidden unit 3. The hidden representations of the 124 training patterns by both networks are depicted in Figure 5.

## 4.3 The Monks 3 problem

The Monks 3 problem turns out to be the easiest of the 3 monks problems to solve. The smallest network that can identify correctly the 6 noisy patterns in the training set is depicted in Figure 6. The first hidden unit is connected to the relevant inputs, while the second hidden unit is not connected to any input. Hence, the connection between the second hidden unit and the output unit allows the output unit to have a nonzero threshold. Since no input is connected to the second hidden unit, its activation value is always  $\sigma(0) = 0.5$ , hence, the output threshold value is equal to the weight of the connection between the second hidden unit and the output unit divided by 2.

The predicted outputs for the noisy patterns were the opposite of the target outputs, and hence an accuracy rate of 95.08 % was obtained on the training set. Because there was no noise in the testing data, the predictive accuracy rate was 100 %. A set of inputs connected to only one hidden unit in the network indicates that the patterns are linearly separable, i.e., there exists a hyperplane such that all monks are on one side of the plane and all non-monks are on the other side. Indeed, when GCA was employed to discretize the activation values of the patterns that have been correctly



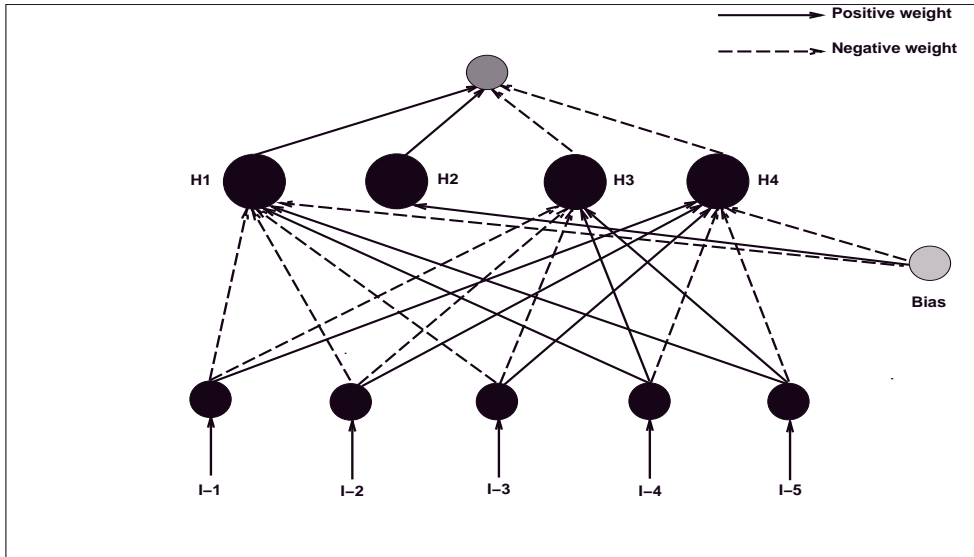
**Figure 6.** A pruned network that solves the Monks 3 problem.

classified by the network, only two unique hidden representations were found, 0 and 100. Hence, the patterns can be divided into two sets, those with activation values in the interval  $[0.0, 1.00)$  and those with activation values of 1.00. Of the 116 non-noisy patterns, 59 patterns are monks and they can be represented by the value 0; the remaining 57 patterns are non-monks and they can be represented by 100.

#### 4.4 The 5-bit parity problem

A network that has been trained and pruned to solve the 5-bit parity problem is shown in Figure 7. Since there are still 4 hidden units left in the network, we run GCA  $4! = 24$  times. The second hidden unit is connected only to the bias input, its activation values are the same for all patterns. The weights of the connections from the bias to the second hidden unit and from this hidden unit to the output unit are 5.3 and 50.96, respectively. Because of its single input connection to the bias, the second hidden unit activation value effectively acts as a nonzero threshold of the output unit. This threshold value is equal to  $50.96 \times \sigma(5.3)$ .

The results from running GCA 24 times produced 4 different hidden representations of the data as depicted in Figure 8. Since the activation values of the second hidden unit are the same for all patterns, we show the representations of the data by the other 3 hidden units. Note that the number of distinct representations is always 6. Let us analyze the representations depicted in the upper left corner of Figure 8. Eight of the 24 GCA runs resulted in these representations. Samples with even and



**Figure 7.** A pruned network that solves the 5-bit parity problem.

odd parity can be distinguished based only on the activation values of the third hidden unit. There are 6 discrete values found by GCA for these activation values, they are 18, 27, 38, 50, 62, and 73. The 6 values correspond to the division of the input space by 6 parallel hyperplanes whose orientations are determined by the weights of the network. These hyperplanes form the boundaries of regions that contain 6 groups of patterns. The first group contains patterns with activation values that fall in the interval  $[0.18, 0.265)$ , the second group in the interval  $[0.265, 0.375)$ , and so on. The inconsistency check of the clustering algorithm ensures that each of these groups contains patterns with the same parity.

It is easy to explain the network prediction in terms of the activation value of an input pattern:

Rule 1: If the activation value is in the interval  $[0.18, 0.265)$ , then **odd** parity.

Rule 2: If the activation value is in the interval  $[0.265, 0.375)$ , then **even** parity.

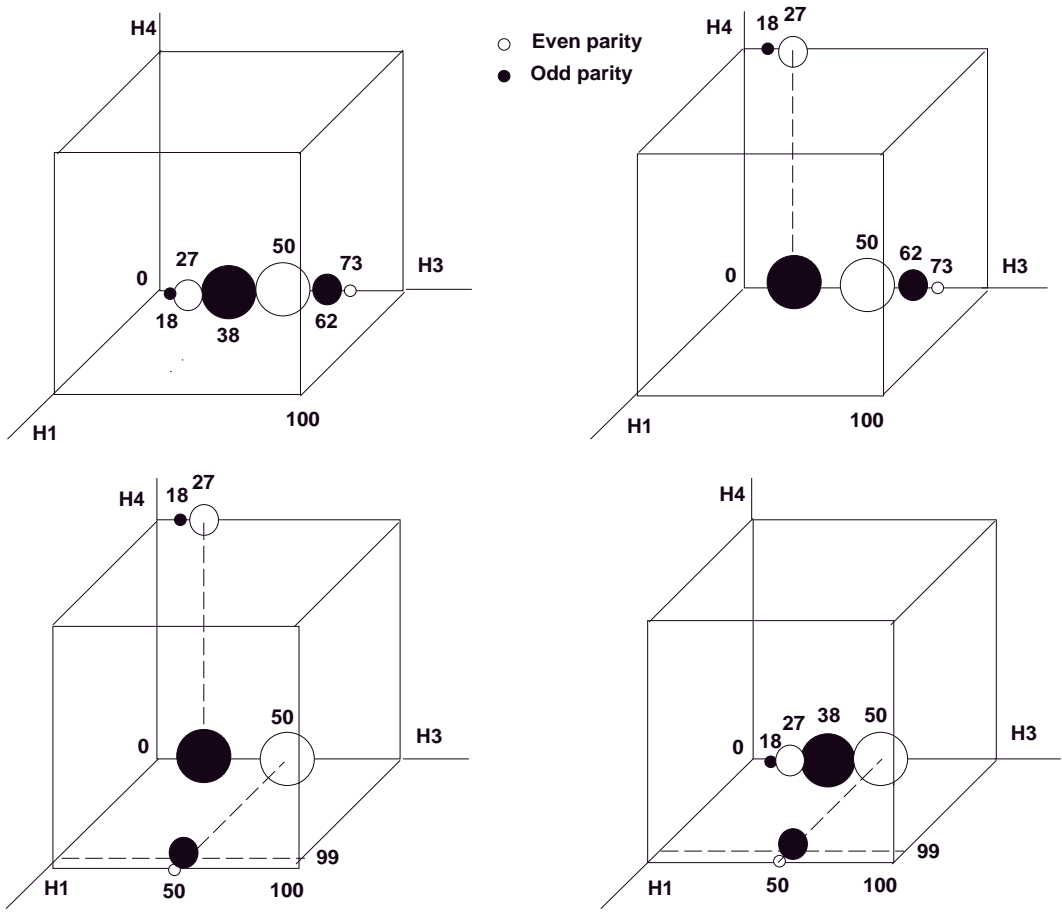
Rule 3: If the activation value is in the interval  $[0.375, 0.495)$ , then **odd** parity.

Rule 4: If the activation value is in the interval  $[0.495, 0.615)$ , then **even** parity.

Rule 5: If the activation value is in the interval  $[0.615, 0.725)$ , then **odd** parity.

Rule 6: If the activation value is in the interval  $[0.725, 1]$ , then **even** parity.

Since there is no pattern with activation value that is less than 0.18, Rule 1 classifies all patterns with activation values less than 0.265 as odd-parity patterns.



**Figure 8.** Discrete representations of the 5-bit parity patterns from the network in Figure 7 by different clustering sequences. Large circle represents 10 patterns, medium 5 patterns, and small 1 pattern. H2 is connected only to the bias, its activation values are constant.

The weights of the connections from inputs 1 to 5 to the third hidden units are -.50, -.50, -.50, 0.51, and 0.51, respectively. An activation value of a pattern will be less than 0.265 if and only if its inputs satisfy the following condition:

$$\sigma(-0.50\mathcal{I}_1 - 0.50\mathcal{I}_2 - 0.50\mathcal{I}_3 + 0.51\mathcal{I}_4 + 0.51\mathcal{I}_5) < 0.265,$$

or equivalently,

$$-0.50\mathcal{I}_1 - 0.50\mathcal{I}_2 - 0.50\mathcal{I}_3 + 0.51\mathcal{I}_4 + 0.51\mathcal{I}_5 < \sigma^{-1}(0.265) = -1.02.$$

By computing the inverse of the sigmoid function for the other 4 values: 0.375, 0.495, 0.615, and 0.725, we obtain the following

Rule 1: If  $-0.50\mathcal{I}_1 - 0.50\mathcal{I}_2 - 0.50\mathcal{I}_3 + 0.51\mathcal{I}_4 + 0.51\mathcal{I}_5 < \sigma^{-1}(0.265) = -1.02$ , then **odd** parity.

Rule 2: If  $-1.02 \leq -0.50\mathcal{I}_1 - 0.50\mathcal{I}_2 - 0.50\mathcal{I}_3 + 0.51\mathcal{I}_4 + 0.51\mathcal{I}_5 < \sigma^{-1}(0.375) = -0.51$ , then **even** parity.

Rule 3: If  $-0.51 \leq -0.50\mathcal{I}_1 - 0.50\mathcal{I}_2 - 0.50\mathcal{I}_3 + 0.51\mathcal{I}_4 + 0.51\mathcal{I}_5 < \sigma^{-1}(0.495) = -0.02$ , then **odd** parity.

Rule 4: If  $-0.02 \leq -0.50\mathcal{I}_1 - 0.50\mathcal{I}_2 - 0.50\mathcal{I}_3 + 0.51\mathcal{I}_4 + 0.51\mathcal{I}_5 < \sigma^{-1}(0.615) = 0.49$ , then **even** parity.

Rule 5: If  $0.49 \leq -0.50\mathcal{I}_1 - 0.50\mathcal{I}_2 - 0.50\mathcal{I}_3 + 0.51\mathcal{I}_4 + 0.51\mathcal{I}_5 < \sigma^{-1}(0.725) = 0.97$ , then **odd** parity.

Rule 6: If  $-0.50\mathcal{I}_1 - 0.50\mathcal{I}_2 - 0.50\mathcal{I}_3 + 0.51\mathcal{I}_4 + 0.51\mathcal{I}_5 \geq 0.97$ , then **even** parity.

Further analysis of these rules reveals interesting clues into the classification process of a neural network. The number of patterns that are classified by Rule  $i, i = 1, 2, \dots, 6$  is exactly the binomial coefficient  $\binom{6}{i}$ . Let us say that there is a *match* between an input and the rule condition if the sign of the coefficient  $\mathcal{I}_k$  is positive and the input  $\mathcal{I}_k = 1$  or if the sign of the coefficient  $\mathcal{I}_k$  is negative and the input  $\mathcal{I}_k = 0$ . Listing all the patterns covered by a rule shows that only patterns with  $i$  matches are classified by Rule  $i + 1$ . Suppose that the number of one's that match is  $i_1$  and the number of zeros that match is  $i_2 = i - i_1$ . Since  $i_1$  one's match and there are 2 positive coefficients in the rule condition, there must be  $2 - i_1$  zero's that do not match. Similarly, since  $i_2$  zero's match, there must be  $3 - i_2$  one's that do not match. The total number of one's in the pattern is then  $i_1 + 3 - i_2$ , which is odd if  $i_1 + i_2 = i$  is even, and even otherwise. Hence, if the total number of one's is

odd, the total number of matches will be even and the pattern will be classified by an odd-numbered rule as odd parity. On the other hand, if the total number of one's is even, a pattern will be classified as even parity by an even-numbered rule.

The concept of even-odd parity can also be learned by noting that the only pattern that is classified by Rule 1 is  $(1, 1, 1, 0, 0)$ . It is classified as odd parity. Each of the 5 patterns classified by Rule 2 is the same as the the first pattern except for 1 bit and they are all classified as odd parity. Exactly 2 of the 5 bits of the 10 patterns classified by Rule 2 are different from those of the pattern classified by Rule 1. These patterns are classified as even parity. Similar observations can be made from patterns classified by the rest of the rules. We can conclude that those patterns with 0, 2 or 4 bits equal to 1 are even parity, while the rest are odd parity.

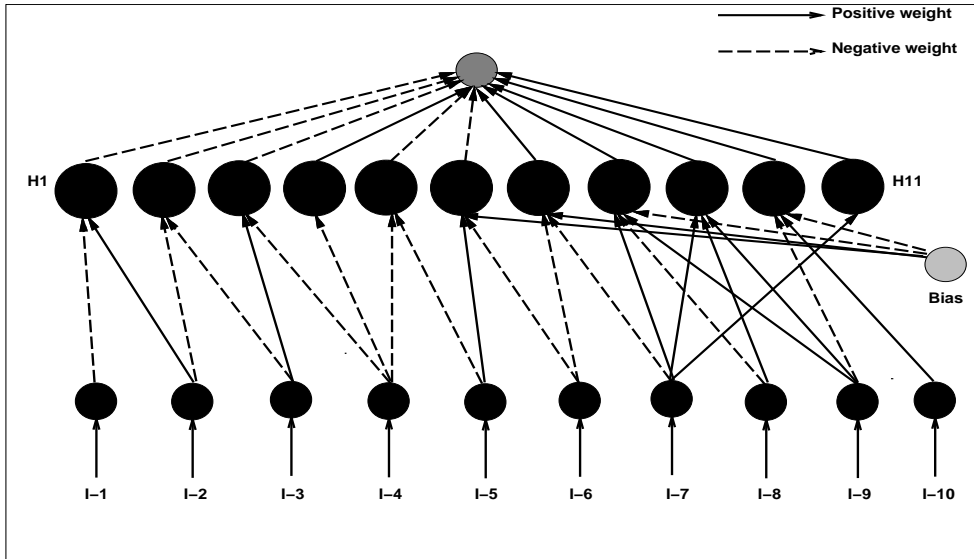
## 4.5 The contiguity problem

We obtained pruned networks with a relatively large number of hidden units for the contiguity problem (Denker *et al.*, 1987; Gorodkin *et al.*, 1993). Each pattern of this problem is a binary string of length 10. If a string contains three consecutive sequences of 1's (clumps), its class label is 1. Otherwise, if a string contains two consecutive sequences of 1's, its class label is 0. Only strings with two or three consecutive sequences of 1's are used as the training patterns. Of the 1024 binary strings of length 10, there are 792 strings with 2 or 3 clumps. One network that correctly classifies all 792 input patterns is depicted in Figure 9. The original network had 20 hidden units. After pruning, 11 hidden units remained.

We clustered the activation values of the hidden units in order, starting from those of H1 and ending with those of H11. The results were rather interesting. At all hidden units, except H11, 2 clusters were found. The activation values at H11 were all clustered into just 1 value. Hence, the pruned network has transformed the 10-dimensional input patterns into 10-dimensional binary hidden representations. However, while there are 792 unique input patterns, there are only 100 unique hidden representations. We shall examine some of these hidden representations.

Since we are interested in each hidden unit's role in representing the input patterns, we focus on individual hidden units. Hidden unit H1 is connected only to inputs  $\mathcal{I}_1$  and  $\mathcal{I}_2$ . To investigate the role of this hidden unit, we group the input patterns as follows:

1. Denote the hidden representation of a pattern by  $\mathcal{A} = \langle \alpha_1, \alpha_2, \dots, \alpha_{11} \rangle$ , where  $\alpha_j = 1$  or 2 (we let  $\alpha_j = 1$  or 2 since there are at most 2 clusters at each hidden unit).
2. Let  $\langle \alpha_1^i, \alpha_2^i, \dots, \alpha_{11}^i \rangle$  be the hidden representation of pattern  $x_i$ .



**Figure 9.** A pruned network with 11 hidden units for the contiguity problem.

3. Find all input patterns with the same hidden representation as  $x_i$ 's, group all these input patterns as the set  $\mathcal{S}$ .
4. If  $\alpha_1^i = 1$ , form set  $\mathcal{S}'$  whose members are all patterns with hidden representations  $\langle 2, \alpha_2^i, \dots, \alpha_{11}^i \rangle$ . Else if  $\alpha_1^i = 2$ , form set  $\mathcal{S}'$  whose members are all patterns with hidden representations  $\langle 1, \alpha_2^i, \dots, \alpha_{11}^i \rangle$ .

There are 28 unique sets  $\mathcal{S}$  with nonempty  $\mathcal{S}'$  in total. The cardinality of these sets ranges from 1 to 69. We show some of the smaller sets below.

1.  $\mathcal{A} = \langle 1, 1, 2, 1, 1, 1, 2, 1, 1, 1, 1 \rangle$ .

The set  $\mathcal{S}$  has only 1 element

$$1 \ 0 \ 1 \ 1 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0$$

while the set  $\mathcal{S}'$  has 3 elements, all of which have hidden representation  $\mathcal{A}' = \langle 2, 1, 2, 1, 1, 1, 2, 1, 1, 1, 1 \rangle$ .

$$0 \ 0 \ 1 \ 1 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0$$

$$0 \ 1 \ 1 \ 1 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0$$

$$1 \ 1 \ 1 \ 1 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0$$

2.  $\mathcal{A} = \langle 1, 1, 2, 2, 2, 1, 2, 1, 1, 1, 1 \rangle$ .

The set  $\mathcal{S}$  has only 1 element

$$1 \ 0 \ 1 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0$$

Five patterns with hidden representation  $\mathcal{A}' = \langle 2, 1, 2, 2, 2, 1, 2, 1, 1, 1, 1 \rangle$  form the set  $\mathcal{S}'$

```

0 0 1 0 0 1 0 0 0 0
0 1 0 0 0 1 0 0 0 0
0 1 1 0 0 1 0 0 0 0
1 1 0 0 0 1 0 0 0 0
1 1 1 0 0 1 0 0 0 0

```

3.  $\mathcal{A} = \langle 1, 2, 1, 1, 1, 2, 1, 2, 2, 1, 1 \rangle$ .

The elements of the set  $\mathcal{S}$  are

```

1 0 0 1 1 1 1 0 1 0
1 0 0 1 1 1 1 0 1 1

```

and the patterns with hidden representation  $\mathcal{A}' = \langle 2, 2, 1, 1, 1, 2, 1, 2, 2, 1, 1 \rangle$  are

```

0 0 0 1 1 1 1 0 1 0
0 0 0 1 1 1 1 0 1 1

```

4.  $\mathcal{A} = \langle 2, 2, 1, 1, 1, 1, 1, 1, 2, 1, 1 \rangle$ .

The elements of the set  $\mathcal{S}$  are

```

0 0 0 1 0 1 1 0 0 0
0 0 0 1 0 1 1 1 0 0
0 0 0 1 0 1 1 1 1 0
0 0 0 1 0 1 1 1 1 1

```

Patterns with hidden representation  $\mathcal{A}' = \langle 1, 2, 1, 1, 1, 1, 1, 1, 2, 1, 1 \rangle$  form the set  $\mathcal{S}'$ :

```

1 0 0 1 0 1 1 0 0 0
1 0 0 1 0 1 1 1 0 0
1 0 0 1 0 1 1 1 1 0
1 0 0 1 0 1 1 1 1 1

```

We notice the following similarities from each pair of sets  $\mathcal{S}$  and  $\mathcal{S}'$ :

1. Patterns in both sets have mostly identical values, inputs  $\mathcal{I}_4 - \mathcal{I}_{10}$  are always the same.
2. All patterns with the hidden unit H1 cluster  $\alpha_1 = 1$  have three clumps, while all patterns with  $\alpha_1 = 2$  have two clumps.

**Table VI.** The 8 input attributes of the Pima Indian diagnosis data set. The minimum and maximum values of each attribute are among those patterns in the training data set.

Input	Description	Min. value	Max. value
$\mathcal{X}_1$	Number of times pregnant	0	14
$\mathcal{X}_2$	Plasma glucose concentration	0	199
$\mathcal{X}_3$	Diastolic blood pressure	0	122
$\mathcal{X}_4$	Triceps skin fold thickness	0	63
$\mathcal{X}_5$	2-hour serum insulin	0	846
$\mathcal{X}_6$	Body mass index	0	67.1
$\mathcal{X}_7$	Diabetes pedigree function	0.84	2.42
$\mathcal{X}_8$	Age	21	72

The above similarities were also observed when we analyzed hidden units H2 to H10. We can say that the hidden units act as local edge detectors. Within a group of almost identical patterns, each hidden unit distinguishes patterns with three clumps from those with two clumps.

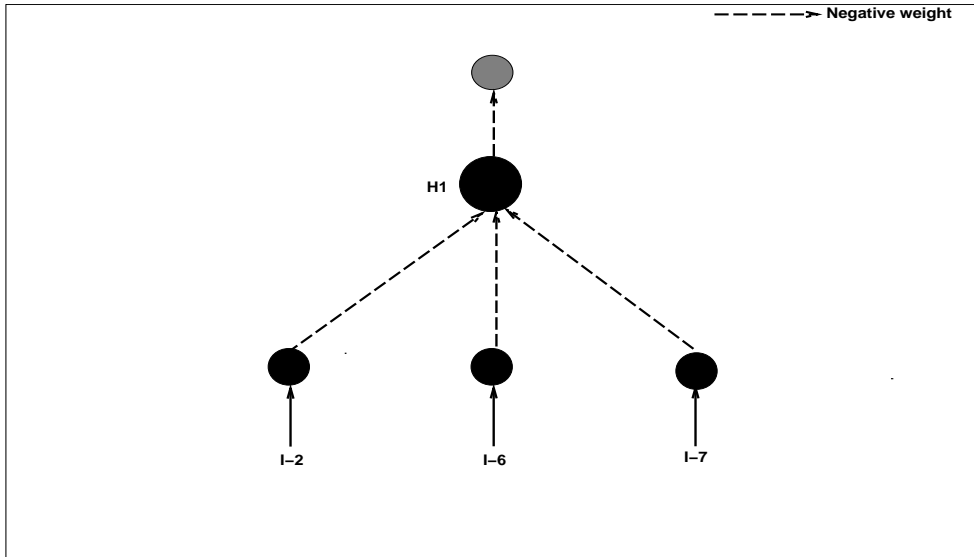
## 5 Experiment using real world data set

We analyze the hidden unit activation values of networks that have been trained to solve the Pima Indian diabetes diagnosis problem. The data set has been obtained via ftp to the machine learning repository at the University of California, Irvine (Merz & Murphy, 1992). The problem is selected because it has been used to test the ARTMAP rule extraction system (Carpenter & Tan, 1995).

The Pima Indian diabetes diagnosis data set consists of 768 patterns, of which 268 are positive patterns from patients diagnosed as diabetic and 500 are negative patterns. Each patterns is described by 8 input features: 1 integer-valued feature and 7 real-valued features (Table VI). We followed the experimental procedure of Carpenter and Tan (1995) and divided the data set into 576 training patterns and 192 testing patterns. All inputs were normalized so that they range in the interval  $[-1, 1]$  using the transformation

$$\mathcal{I}_i = 2 \left( \frac{\mathcal{X}_i - \mathcal{X}_{i,min}}{\mathcal{X}_{i,max} - \mathcal{X}_{i,min}} \right) - 1, \quad (5)$$

where  $\mathcal{X}_{i,min}$  and  $\mathcal{X}_{i,max}$  are the minimum and maximum values of the attribute  $\mathcal{X}_i$ , respectively.



**Figure 10.** A pruned network for the Pima Indian diabetes diagnosis problem. The network predicts correctly 76.39 % of the training patterns and 82.81 % of the testing patterns.

Including the input for hidden unit bias, a total of 9 input units were needed. Thirty neural networks each with 4 hidden units were trained and pruned. Aiming to achieve similar accuracy level as those reported by Carpenter and Tan, the pruning process was terminated as soon as the accuracy of the network on the training data set dropped below 75 %. The average number of connections of the 30 pruned network was 11.27 and the average accuracy rates on the training and testing data set were 76.36 % and 78.28 %, respectively.

One of the networks having the fewest number of connections is depicted in Figure 10. It has only 1 hidden unit and 4 connections left. It correctly predicts 440 (76.39 %) of the 576 training patterns and 159 (82.81 %) of the 192 testing patterns. When we applied GCA to the hidden unit activation values of the patterns that had been correctly classified by the pruned network, 2 clusters were found. All activation values in the subinterval  $[-0.75, 0.01)$  formed one cluster and those in  $[0.01, 1]$  formed the other. There was no pattern with activation value less than -0.75. All 121 patterns with activation values in the first subinterval are positive patterns, while the 319 patterns with activation values in the second subinterval are negative patterns. Hence, we obtain the simple rule:

Rule 1. If the activation value is in  $[-0.75, 0.01)$ , then **positive**.

Default rule: **negative**.

The weights of the connections from the input units  $\mathcal{I}_2, \mathcal{I}_6$  and  $\mathcal{I}_7$  are -0.47, -0.40 and -0.28, respectively. It follows that an activation value of a pattern will be less than 0.01 if and only if

$$\delta(-0.47\mathcal{I}_2 - 0.40\mathcal{I}_6 - 0.28\mathcal{I}_7) < 0.01$$

Since  $\delta(0.01) = 0.01$ , the above condition is equivalent to

$$-0.47\mathcal{I}_2 - 0.40\mathcal{I}_6 - 0.28\mathcal{I}_7 < 0.01$$

By making use of the relationship between the transformed inputs  $\mathcal{I}_i$  and the original inputs  $\mathcal{X}_i$  given by equation (5), we obtain the following rule:

Rule 1. If  $\mathcal{X}_2 + 2.51\mathcal{X}_6 + 50.10\mathcal{X}_7 > 246.16$ , then **positive**.

Default rule: **negative**.

The accuracy rates of this rule on the training and testing data sets are identical to those of the pruned network, 76.39 % and 82.81 %, respectively. In contrast, the rule sets extracted by the ARTMAP system contain up to 28 fuzzy rules involving all 8 input attributes of the data and their best predictive accuracy rate is 79 % (Carpenter & Tan, 1995). We also run C4.5 for comparison. The accuracy rate of the rules extracted by C4.5 is 85.9 % on the training data set. However, on the testing data set the accuracy rate is only 79.2 %. As C4.5 generates rules that are axis-parallel, many rules are needed to separate the positive and negative patterns. Twelve rules are generated by C4.5 and some rules have 10 antecedents involving 5 attributes of the data.

## 6 Discussion

A main selling point of the neural network approach for pattern classification has been its accuracy. Compared with the decision tree methods, neural networks can obtain better accuracy rates. This higher accuracy is often achieved in the presence of noise or of some missing values in the data. By analyzing the hidden representations of a network, it is also possible to generate rules that explain the network decision process. For problems such as the Monks 2 problem, the network rules are not only more accurate than the decision tree rules, but they are also more concise. The better performance of the networks, in terms of both accuracy and compactness of the extracted rules, can be explained by the inherent differences between the two approaches:

1. Neural networks consider the effect of the inputs on the class label together, while most decision trees consider only one input at a time. In the Monks 2 example, a single rule involving all 6 relevant inputs was obtained from a pruned network. In contrast to this result, C4.5 generated 10 rules, each involving not more than 4 input attributes.
2. Decision trees methods that consider only one input at a branching node divide the patterns' space by planes that are axis parallel, neural networks, on the other hand, are capable of generating planes with an arbitrary orientation.
3. Neural networks are trained with the whole data set. In contrast, decision tree methods use smaller and smaller subsets of the data as a tree is generated; this can lead to overfitting of the data and poor generalization.
4. By using data consistency as the criterion for clustering the activation values of the hidden units, some hidden units of a network can be removed. If symbolic rules are to be generated from the network, fewer hidden units will result in more concise rules.
5. Most decision trees are binary trees, at each node a condition is checked whether it is satisfied or not satisfied. Each hidden unit of a network, on the other hand, can be thought of as a branching node that leads to two or more child-nodes.

## 7 Conclusion

The hidden layer of a backpropagation network holds the key to understanding neural network learning. We presented a simple greedy clustering to analyze the activation values of a trained network. By applying this algorithm to networks that have been trained to solve problems with known concepts, we gained better insights to the classification process of the networks, such as why neural networks can achieve higher accuracy than decision trees. If desired, the classification process of the networks can be restated as sets of explicit decision rules.

Our experimental results show that the hidden unit activation values of networks that have been trained on large data sets can also be represented by a few clusters. The rules obtained from analyzing the hidden representations achieve accuracy rates comparable to those extracted by the ARTMAP rule extraction system. However, the former are far more concise than the latter.

## Acknowledgements

The authors wish to thank two anonymous reviewers whose suggestions and comments greatly improve the presentation of this paper.

## References

- Andrews, R., Diederich, J. & Tickle, A.B. (1995) Survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge-Based Systems*, **8(6)**, 373–389.
- Blassig, R. (1994) GDS: Gradient descent generation of symbolic classification rules. In *Advances in Neural Information Processing Systems*, Vol. 6, 1093–1100. San Mateo, CA: Morgan Kaufmann.
- Carpenter, G.A. & Tan, A-H. Rule extraction: From neural architecture to symbolic representation. *Connection Science*, **7(1)**, 3–28.
- Denker, J., Schwartz, D., Wittner, S., Solla, S.A., Howard, R., Jackel, L. & Hopfield, J. (1987) Large automatic learning, rule extraction and generalization. *Complex System*, **1**, 877–922.
- Dietterich, T.G., Hild, H. & Bakiri, G. (1990) A comparative study of ID3 and backpropagation for english text-to-speech mapping. In *Machine Learning: Proceedings of the Seventh International Conference*. University of Texas, Austin, Texas.
- Fisher, D.H. & McKusick, K.B. (1989) Empirical comparison of ID3 and backpropagation. *Proceedings of 11th International Joint Conference on Artificial Intelligence*. Menlo Park, CA: AAAI Press, 788–793.
- Fu, L. (1991) Rule learning by searching on adapted nets. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, Menlo Park, CA: AAAI Press/The MIT Press, 590–595.
- Gorodkin, J., Hansen, L.K., Krogh, A. & Svarer, C. (1993) A quantitative study of pruning by optimal brain damage. *Int. Journal of Neural Systems*, **4(2)**, 159–169.
- Lang, K.J. & Witbrock, M.J. (1988) Learning to tell two spirals apart. In *Proceedings of the 1988 Connectionist Summer School*, San Mateo, CA: Morgan Kaufmann, 52–59.

- Merz, C.J. & Murphy, P.M. (1996) UCI repository of machine learning databases [machine readable data repository]. Irvine, CA: University of California, Department of Information and Computer Science.
- Quinlan, J.R. (1993) *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann.
- Quinlan, J.R. (1994) Comparing connectionist and symbolic learning methods. In S.J. Hanson, G.A. Drastall, and R.L. Rivest (Eds), *Computational Learning Theory and Natural Learning Systems*, volume 1, A Bradford Book, The MIT Press, 445–456.
- Setiono, R. (1995) A neural network construction algorithm which maximizes the likelihood function. *Connection Science*, **7(2)**, 147–166.
- Setiono, R. (1997) A penalty-function approach for pruning feedforward neural networks. *Neural Computation*, **9(1)**, 301–320.
- Setiono, R. & Liu, H. (1996) Symbolic representation of neural networks. *Computer*, March 1996, pp. 71–77.
- Setiono, R. & Liu, H. (1995) Understanding Neural Networks via Rule Extraction. In the *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, 480–485.
- Shavlik, J.W., Mooney, R.J., & Towell, G.G. (1991) Symbolic and neural learning algorithms: An experimental comparison. *Machine Learning*, **6(2)**, 111–143.
- Thrun, S.B., et al. (1991) The MONK's problems - a performance comparison of different learning algorithm. Preprint CMU-CS-91-197, Carnegie Mellon University, Pittsburgh, PA.
- Towell, G.G. & Shavlik, J.W. (1993) Extracting refined rules from knowledge-based neural networks. *Machine Learning*, **13(1)**, 71–101.
- van Ooyen, A. & Nienhuis, B. (1992) Improving the convergence of the backpropagation algorithm. *Neural Networks*, **5**, 465–471.
- Weiss, S.M. & Kapouleas, I. (1989) An empirical comparison of pattern recognition, neural nets, and machine learning classification methods. In *Proceedings of 11th International Joint Conference on Artificial Intelligence*, 781–787.