When local isn't enough: Extracting distributed rules from networks

David A. Medler
Center for the Neural Basis of Cognition
Carnegie Mellon University
medler@cnbc.cmu.edu

David B. McCaughan
Department of Computer Science
University of Alberta
davidb@cs.ualberta.ca

Michael R. W. Dawson & Leanne Willson
Biological Computation Project, Department of Psychology
University of Alberta
mike@bcp.psych.ualberta.ca
lwillson@bcp.psych.ualberta.ca

Abstract

Recent work on a network interpretation technique called "banding analysis" has shown how symbolic rules can be easily extracted from trained connectionist networks [2]. This method, however, is limited to "local" interpretations, and cannot take full advantage of the distributed nature of neural networks. In this paper we discuss a technique for extracting distributed symbolic rules from trained connectionist networks. This technique is based on the k-means cluster analysis of activation values across all hidden units instead of single units as in the banding analysis. A novel stopping rule—based on domain specific heuristic information—is defined such that the optimum number of clusters are extracted from the network. Once the appropriate clusters are determined, simple statistics are computed to determine the shared characteristics of all patterns within a cluster. The utility of this technique is illustrated on two benchmark problems. It is concluded that this form of distributed rule extraction is more succinct than simple local analysis of internal structure.

Introduction

One of the most challenging problems facing connectionist researchers is the interpretation of the internal structure of trained neural networks; that is, the extraction of "rules" from the network in order to understand how the network is solving a particular problem. Such extraction techniques are beneficial to the development of expert systems and data mining techniques based on neural networks.

There are two main approaches to network interpretation: analysis of network topology (i.e., connection weights and unit biases [4]), and analysis of network behavior in terms of either input/output mappings [10] or internal activation states [2]. The first interpretation technique gives a static picture of how a network solves a problem, whereas the second technique gives a more dynamical view. In this paper, we will focus on this second form of analysis; specifically, this work is based on the analysis of hidden unit activations distributed throughout a network.

In [2] it was shown how a network interpretation technique—dubbed "banding analysis"—could extract symbolic rules from a particular type of connectionist network using non-monotonic processing units. This technique depends upon the analysis of local properties of individual hidden units; When plotted with a jittered density plot, hidden unit activations of non-monotonic units [2] [5] are arranged into distinct and interpretable bands. We show in [6] that this banding reflects sets of points lying in hyperplanes perpendicular to the weight vector of a given unit. The analysis of these bands, however, only reveals the local structure within each hidden unit.

It is possible that these local bands may merely be an artifact of more general regularities that are distributed across a number of different hidden units. In other words, the local analysis may reveal structures that are inconsequential for solving the problem, whereas a more distributed analysis may be able to extract those properties actually required for solving the problem. For instance, many of the "bands" identified by the local analysis often contain "micro-bands" that share very few (if any) common features when grouped together.

In such cases, variations of more traditional analytic techniques, such as cluster analysis, may be more appropriate to use than banding analysis. We show here that distributed symbolic rules can be extracted from connectionist networks by means of a cluster analysis on hidden unit activations, and that the local rules extracted by banding analysis may be insufficient for overall network interpretation.

In this paper, we introduce a distributed rule extraction technique based on the analysis of hidden unit activations using a cluster analysis with a novel stopping rule. We then illustrate the utility of this technique on two benchmark problems, with a brief comparison between local and distributed analysis techniques. It is concluded that this distributed analysis is both simpler and more complete than local analyses as it reveals regularities across network units often hidden by local analyses.

Method

Analysis Technique

Both the local and distributed approaches to network analysis begin by recording the activites produced by each input pattern within each of the hidden units. This step produces an $m \times n$ data matrix, where m is the number of patterns and n is the number of hidden units.

For local analyses, each column is plotted in a jittered density plot (e.g., see Figure 3), and simple statistics (i.e., means, standard deviations, and pair-wise correlations) are performed on the patterns falling into each "band" defined by the plots (see [6]). This local interpretation is limited by two factors: (i) the identification of unique bands is qualitative in nature, and (ii) it is blind to the regularities in the patterns of activities distributed across units.

However, we can use these activity patterns for a distributed analysis of the network. Specifically, we can extract distributed rules from a network by performing a k-means cluster analysis on the vectors of hidden unit activities. The k-means cluster analysis is an iterative process that assigns each pattern to one of k specific clusters such that the assigned pattern is closer to the centroid of that cluster than to any other cluster.

One problem inherent to cluster analysis is determining the number of clusters, k, to use. Normally, one starts with a fairly small number of clusters, and then adds clusters until a suitable number is found as determined by a predefined "stopping rule". Here we propose a novel stopping rule specific to neural networks. As the network has already found a solution to the problem, we define the new stopping rule as the minimum number of clusters required such that all patterns within a single cluster produce the same network output.

As with local band interpretation, we can then perform a definite feature analysis [2] to uncover common characteristics of the input patterns corresponding to the activity vectors in each cluster. These definite features can then be used to create a symbolic rule base.

Problem Types

Two different bench-mark problems were considered: the first of the "*3-Monks*" problem [9], and the "*Mushroom*" problem [8]. The first of the *3-Monks* problem consists of 124 training patterns and 432 test patterns defined by six attributes ($a_1 \rightarrow a_6$; see Table 1). The network is trained on the target concept (head-shape = body-shape) OR (jacket-color = red) [i.e., $(a_1 = a_2) \vee (a_5 = 1)$]. The network inputs were encoded using a unary encoding scheme such that only one input unit for each characteristic—as indicated by the input groupings in Figure 1—was active at any one time.

Table 1. Monk Attributes

	ATTRIBUTE	VALUE
a_1	head-shape	[1]round, [2]square, [3]octagon
a_2	body-shape	[1]round, [2]square, [3]octagon
a_3	is-smiling	[1]yes, [2]no
a_4	holding	[1]sword, [2]balloon, [3]flag
a_5	jacket-color	[1]red, [2]yellow, [3]green, [4]blue
a_6	has-tie	[1]yes, [2]no

Table 2. Mushroom Attributes

	ATTRIBUTE	VALUE
a_1	cap-shape	[1]bell, [2]conical, [3]convex, [4]flat, [5]knobbed, [6]sunken
a_2	cap-surface	[1]fibrous, [2]grooves, [3] scaly, [4]smooth
a_3	cap-color	[1]brown, [2]buff, [3]cinnamon, [4]gray, [5]green, [6]pink, [7]purple, [8]red, [9]white, [10]yellow
a_4	bruises	[1]no, [2]yes
a_5	odor	[1]almond, [2]anise, [3]creosote, [4]fishy, [5]foul, [6]musty, [7]none, [8]pungent, [9]spicy
a_6	gill-attachment	[1]attached, [2]descending, [3]free, [4]notched
a_7	gill-spacing	[1]close, [2]crowded, [3]distant
a_8	gill-size	[1]broad, [2]narrow
a_9	gill-color	[1]black, [2]brown, [3]buff, [4]chocolate, [5]gray, [6]green, [7]orange, [8]pink, [9]purple, [10]red, [11]white, [12]yellow
a_{10}	stalk-shape	[1]enlarging, [2]tapering
a_{11}	stalk-surface-above-ring	[1]fibrous, [2]scaly, [3]silky, [4]smooth
a_{12}	stalk-surface-below-ring	[1]fibrous, [2]scaly, [3]silky, [4]smooth
a_{13}	stalk-color-above-ring	[1]brown, [2]buff, [3]cinnamon, [4]gray, [5]orange, [6]pink, [7]red, [8]white, [9]yellow
a_{14}	stalk-color-below-ring	[1]brown, [2]buff, [3]cinnamon, [4]gray, [5]orange, [6]pink, [7]red, [8]white, [9]yellow
a_{15}	veil-type	[1]partial, [2]universal
a_{16}	veil-color	[1]brown, [2]orange, [3]white, [4]yellow
a_{17}	ring-number	[1]none, [2]one, [3]two
a_{18}	ring-type	[1]cobwebby, [2]evanescent, [3]flaring, [4]large, [5]none, [6]pendant, [7]sheathing, [8]zone,
a_{19}	spore-print-color	[1]black, [2]brown, [3]buff, [4]chocolate, [5]green, [6]orange, [7]purple, [8]white, [9]yellow
a_{20}	population	[1]abundant, [2]clustered, [3]numerous, [4]scattered, [5]several, [6]solitary
a_{21}	habitat	[1]grasses, [2]leaves, [3]meadows [4]paths, [5]urban, [6]wastes, [7]woods

The *Mushroom* problem consists of the hypothetical description of 8124 different mushrooms, each characterized by 21 different multi-featured properties; the network's goal is to classify the mushrooms as edible (4208 instances) or poisonous (3916 instances). Two modifications were made to the original data set. First, one characteristic (stalk-root) that had missing information was removed from the data set as it was not essential for classification. Second, the data set was expanded by adding extra output encodings corresponding to previous "rules" as determined by prior processing (see [7] and Figure 4). Therefore, the network was actually trained to produce both a classification and a "reason" for the classification [1].

As the mushroom characteristics were multi-featural, the features were represented by discrete, real numbers between 0 and 1. That is, if a characteristic had five features, the features were represented as the values 0.00, 0.25, 0.50, 0.75, and 1.00. For later analysis, the input features were converted into a 119 unary code; again, only one feature was active at any one time within each characteristic group.

Network Architecture and Training

Value unit networks (networks with non-monotonic processing units [3]) trained with a modified generalized delta rule were trained for both problems. For the *3-Monks* problem, the network consisted of 17 input units, 2 hidden units, and 1 output unit with full connectivity between layers (see Figure 1). Network weights were randomized between ± 0.1 and all biases were initialized to 0.0. A learning rate of 0.1 with no momentum was used; the network was trained to a tolerance level of 0.01 (meaning a value of 0.9 or higher when the target is 1.0, and 0.1 or lower when the target is 0.0).

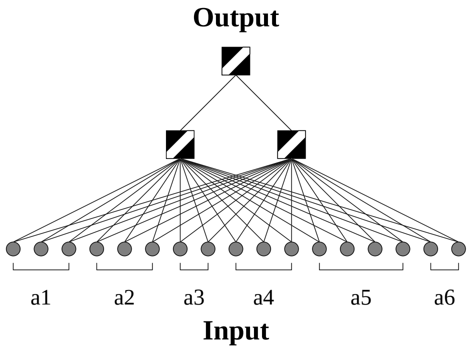

Figure 1. Network architecture for the 3-Monks problem. Note that unary encoding is being used.

For the *Mushroom* problem, the network consisted of 21 input units, 5 hidden units, and 10 output units. One output unit was used to indicate edible or poisonous, and the other 9 units were used to represent the classification rule (only one of these nine units was on at any time; see Figure 2). For the *Mushroom* problem, a learning rate of 0.005 with no momentum was used. Weights were randomized between ± 1.0 and biases were initialized to zero. The network was trained to a tolerance of 0.0025.

[1]It should be noted that this elaboration is not necessary for the network to solve the problem.

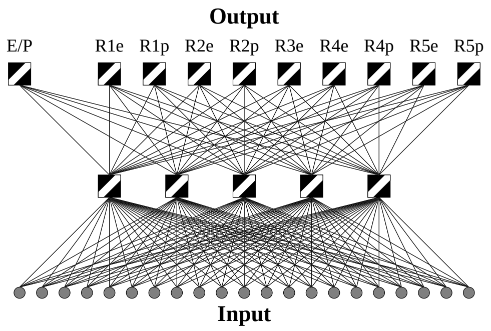

Figure 2. Network architecture for the Mushroom problem. The leftmost output unit, labeled "E/P", was trained to classify the mushroom as either edible or poisonous. The remaining 9 output units were trained to represent the rule used to classify the mushroom (e.g., "R1e" for "Rule 1, edible", "R1p" for "Rule 1, poisonous", etc.).

Results

The 3-Monks *Problem*

The network trained on the *3-Monks* problem converged on a solution in 433 epochs (where an epoch is defined as the full presentations of the data set). The trained network was also able to correctly classify all test patterns. For comparison, both the local and distributed methods of rule extraction are performed on this network.

Banding Analysis From the local banding analysis, six bands could be identified across the two hidden units; the individual bands are labeled alphabetically (clusters are labeled numerically for comparison) in Figure 3 and their interpretations are given in Table 3.

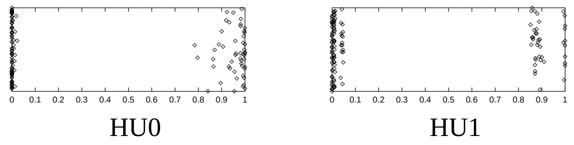

Figure 3. The jittered density plots for the two hidden units. Local bands within each unit are labeled alphabetically. Distributed clusters are labeled numerically.

It should be noted that definite feature analysis is based on individual input units; for the sake of clarity, however, the individual units have been collapsed into their respective attributes. For example, in H0-B (Hidden Unit 0; Band B), the correlation between input unit 3 (i_3) and input unit 6 (i_6) was -1. This can be logically collapsed into the expression "$a_1 \neq a_2$" as the unary encoding precludes more than one value being active within an attribute. The other bands are similarly interpreted.

Table 3. Band interpretation for the *3-Monks* problem.

UNIT	BAND	PATS	INTERPRETATION
H0	A	73	No definite features
	B	51	$a_1 \neq a_2 [i_3 \neq i_6]; a_5 \neq 1$
H1	A	68	No definite features
	B	16	$a_3 = 1; a_2 = 3; a_5 \neq 1$
	C	29	$a_1 \neq a_2 \{i_1, i_4 = 0; i_2 \neq i_3\}; a_5 \neq 1$
	D	11	$a_1 \neq a_2 \{i_3, i_6 = 0; i_1 \neq i_4\}; a_5 \neq 1$

The interpretation of each band follows. Bands H0-A and H1-A have no definite features (this is typical of bands near 0.0; see [6]). All other bands have a unique encoding of "head-shape is not equal to body-shape" as well as the common interpretation "jacket-color is not red." Band H0-B picks out those patterns in which either "head-shape is round" or "body-shape is round." Band H1-B picks out those patterns with attributes "head-shape is round, body-shape is octagon." Bands H1-C and H1-D are somewhat more complicated: H1-C detects "head-shape is round, body-shape is square" or "head-shape is square, body-shape is round" whereas H1-D picks out "head-shape is square, body-shape is octagon" or "head-shape is octagon, body-shape is square."

Consequently, from the local analysis, it appears the network is solving the logical negation of the problem. That is, the network only produces a correct response[2] if it doesn't detect any of the invalid patterns.

Cluster Analysis Although the local band analysis gives an idea as to the role of each hidden unit in solving the problem, it is unclear how these units combine their resources. Cluster analysis on the pattern of activations across units, however, gives you this information explicitly.

Table 4 lists the four clusters (number of patterns, output activation state, mean, and standard deviation of each cluster) extracted from the network and Table 5 lists the interpretations of each cluster. Clusters are labeled numerically in Figure 3.

Similar to the local banding analysis, Cluster 1 (C-1) contains no definite features; however, from the cluster analysis, we know that all patterns in C-1 are valid patterns. The three other clusters are variations on "head-shape is not equal to

[2]The bias of the output unit was +0.008, which means the maximum activation value of 1 is produced when the summed inputs are near zero.

Table 4. The four clusters extracted for the 3-Monks problem. With each cluster is listed the number of patterns in the cluster, the output state of the network, and the mean activity (and standard deviation) of the cluster in each hidden unit.

CLUSTER	PATS	OUT	HU0	HU1
1	62	1	0.002 (0.005)	0.005 (0.005)
2	29	0	0.948 (0.059)	0.876 (0.014)
3	11	0	0.000 (0.000)	0.998 (0.003)
4	22	0	0.963 (0.046)	0.032 (0.019)

body-shape" and "jacket-color is not red". In fact, it is clear that the three clusters exhaustively encode the three, specific exclusive relationships. That is, C-2 encodes "round vs. square", C-3 encodes "square vs. octagon", and C-4 encodes "round vs. octagon." By collapsing across the last three clusters, we extract the symbolic rule "head-shape $\neq$ body-shape AND jacket-color $\neq$ red" which is the logical negation of "head-shape = body-shape OR jacket-color = red".

Table 5. Cluster analysis for the 3-Monks problem.

CLUSTER	INTERPRETATION
1	No definite features
2	$a_1 \neq a_2 \{i_1, i_4 = 0\}; a_5 \neq 1$
3	$a_1 \neq a_2 \{i_3, i_6 = 0\}; a_5 \neq 1$
4	$a_1 \neq a_2 \{i_2, i_5 = 0\}; a_5 \neq 1$

In comparison to the banding analysis, the cluster analysis is simpler and cleaner. Furthermore, as these clusters are distributed across the units, we also explicitly know how the network is solving the problem.

The Mushroom Problem

In this section, we show how this technique can be successfully used on a much larger problem—the *Mushroom* problem contains more patterns (8124), attributes (21), and possible values (up to 12). Furthermore, the decision rules to correctly classify all mushrooms are more complicated (e.g., see Figure 4 for one possible solution to the problem). The network was able to converge on the correct solution after 8699 epochs. As before, the first step in the analysis is to record the hidden unit activations produced by each of the input patterns.

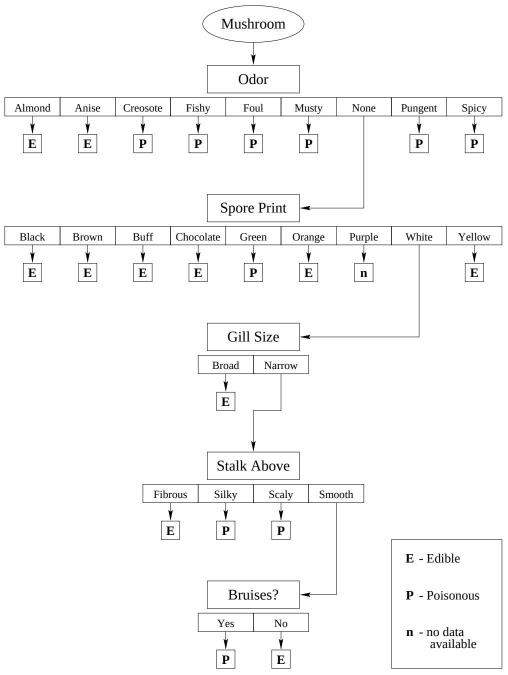

Figure 4. One possible decision tree for solving the mushroom problem. This solution was extracted using a modification of the ID3 algorithm [7].

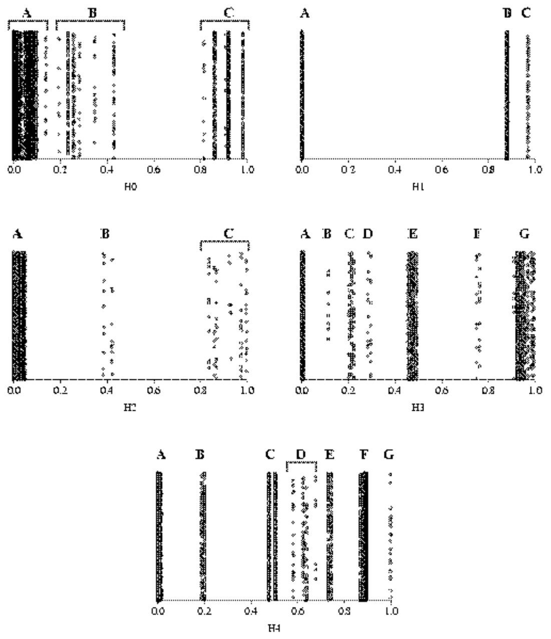

Figure 5. Jittered density plots of the activation values in the five hidden units for the mushroom problem. Local bands as determined by previous analyses are shown.

The hidden unit activations are shown in Figure 5. A previous banding analysis has been performed on this network [7], and these bands are labeled alphabetically within the figure; however, as there are 40 distinct bands across the five hidden units, the current results are restricted to a distributed cluster analysis only.

As opposed to the 40 bands identified in Figure 5, the cluster analysis identifies 12 clusters; these clusters and their accompanying statistics are listed in Table 6. The interpretations for each of the clusters are listed in Table 7. A quick caveat must be made—many of the clusters contain other definite features than those listed, but these irrelvant features (e.g., features constant across all patterns) can be removed by a simple redundancy reduction algorithm. The interpretations listed here have had this algorithm applied.

By moving through the table from the simplest to most complex, Table 7 provides a simple, algorithmic method for classifying mushrooms. In fact, these clusters can be directly translated into the decision tree shown in Figure 4. That is, we are able to easily extract an interpretable, symbolic rule base from a trained neural network. As the interpretation is based on the activation values distributed across units, the rules extracted from the network are complete, as opposed to being sensitive to local structure that may be irrelevant to solving the problem (for such an analysis, see [7, 1]).

Table 6. The 12 clusters extracted for the mushroom problem. Listed with each cluster is the number of patterns, the network output state, and the mean (and standard deviation) associated with the cluster.

CLS	PATS	OUT	HU0	HU1	HU2	HU3	HU4
1	3796	R1p	0.046 (0.031)	0.000 (0.000)	0.012 (0.015)	0.001 (0.002)	0.001 (0.003)
2	704	R1e	0.903 (0.026)	0.879 (0.002)	0.000 (0.000)	0.602 (0.210)	0.409 (0.132)
3	96	R1e	0.982 (0.001)	0.000 (0.000)	0.000 (0.000)	0.213 (0.007)	0.000 (0.000)
4	528	R3e	0.000 (0.000)	0.000 (0.001)	0.017 (0.022)	0.942 (0.026)	0.000 (0.002)
5	40	R4p	0.290 (0.058)	0.000 (0.000)	0.974 (0.026)	0.001 (0.001)	0.645 (0.016)
6	72	R2p	0.000 (0.000)	0.969 (0.001)	0.000 (0.000)	0.927 (0.005)	0.000 (0.000)
7	12	R4e	0.081 (0.000)	0.000 (0.000)	0.421 (0.001)	0.762 (0.000)	0.581 (0.000)
8	12	R5e	0.910 (0.001)	0.000 (0.000)	0.868 (0.001)	0.748 (0.000)	0.579 (0.000)
9	2832	R2e	0.025 (0.081)	0.000 (0.000)	0.000 (0.000)	0.001 (0.001)	0.872 (0.050)
10	8	R5p	0.001 (0.000)	0.000 (0.000)	0.860 (0.001)	0.115 (0.000)	0.000 (0.000)
11	12	R4e	0.050 (0.000)	0.000 (0.000)	0.385 (0.001)	0.284 (0.000)	0.998 (0.000)
12	12	R5e	0.814 (0.001)	0.000 (0.000)	0.836 (0.001)	0.295 (0.000)	0.998 (0.000)

Table 7. Interpretation of the 12 clusters for the *Mushroom* problem. Note that the clusters are listed in order of complexity.

Cluster	Interpretation
2	If ($a_5 = 1 \vee 2$) then edible
3	If ($a_5 = 1 \vee 2$) then edible
1	If ($a_5 = 3 \vee 4 \vee 5 \vee 6 \vee 8 \vee 9$) then poisonous
9	If ($a_5 = 7$) and ($a_{19} = 1 \vee 2 \vee 3 \vee 4 \vee 6 \vee 9$) then edible
6	If ($a_5 = 7$) and ($a_{19} = 5 \vee 7$) then poisonous
4	If ($a_5 = 7$) and ($a_{19} = 8$) and ($a_8 = 1$) then edible
7	If ($a_5 = 7$) and ($a_{19} = 8$) and ($a_8 = 2$) and ($a_{11} = 1$) then edible
11	If ($a_5 = 7$) and ($a_{19} = 8$) and ($a_8 = 2$) and ($a_{11} = 1$) then edible
5	If ($a_5 = 7$) and ($a_{19} = 8$) and ($a_8 = 2$) and ($a_{11} = 2 \vee 3$) then poisonous
8	If ($a_5 = 7$) and ($a_{19} = 8$) and ($a_8 = 2$) and ($a_{11} = 4$) and ($a_4 = 1$) then edible
12	If ($a_5 = 7$) and ($a_{19} = 8$) and ($a_8 = 2$) and ($a_{11} = 4$) and ($a_4 = 1$) then edible
10	If ($a_5 = 7$) and ($a_{19} = 8$) and ($a_8 = 2$) and ($a_{11} = 4$) and ($a_4 = 2$) then poisonous

Discussion

Networks with non-monotonic processing units tend to develop hidden unit representations that are very regular due to the positioning of hyperplanes (defined by the weight vectors) within the problem space [6]. Although individual units lend themselves to a local "banding" analysis, we can exploit the distributed nature of connectionist networks by performing a k-means cluster analysis on the activation values within all hidden units of the network. This clustering technique is dependent on a novel stopping rule based upon trained network responses; that is, all patterns falling within a cluster will produce the same network output. Thus, the interpretation of such clusters provides a unique solution to how the network is solving the problem.

The clusters extracted from the networks also tend to have a simpler interpretable structure when compared to the local banding analysis. Whereas the local banding analysis picks out all related properties of input patterns falling within a single hyperplane, the cluster analysis picks out the subset of properties shared by patterns falling at the intersection of two or more hyperplanes. In other words, cluster analysis provides a more succinct interpretation by extracting only those properties relevant to solving the problem.

Importantly, the related activation values extracted by the cluster analysis can be readily converted into a symbolic rule base. Furthermore, as the clusters are based on the activation values across all units, these rules will give a complete description of how the network is solving the problem. That is, we can take full advantage of the distributed nature of neural networks without sacrificing clear interpretation. This interpretation is a critical step in understanding how connectionist networks solve problems.

References

[1] M. R. W. Dawson and D. A. Medler. Of mushrooms and machine learning: Identifying algorithms in a PDP network. *Canadian Artificial Intelligence*, 38:14–17, 1996. 305v

[2] M. R. W. Dawson, D. A. Medler, and I. S. N. Berkeley. PDP networks can provide symbolic models that are not mere implementations of classical theories. *Philosophical Psychology*, 10:25–40, 1997. 305i, 305i, 305i, 305i, 305ii

[3] M. R. W. Dawson and D. P. Schopflocher. Modifying the generalized delta rule to train networks of non-monotonic processors for pattern classification. *Connection Science*, 4:19–31, 1992. 305iii

[4] M. J. Healy. A topological semantics for rule extraction with neural networks. *Connection Science*, 11:91–113, 1999. 305i

[5] D. B. McCaughan. On the properties of periodic perceptrons. In *Proceedings of the 1997 International Conference on Neural Networks*, pages 188–193, Houston, TX, 1997. 305i

[6] D. B. McCaughan, D. A. Medler, and M. R. W. Dawson. Internal representation in networks of non-monotonic processing units. In *Proceedings of the 1999 International Conference on Neural Networks*, pages 304i–304vi, Washington, DC, 1999. 305i, 305ii, 305iv, 305vi

[7] D. A. Medler. *The Crossroads of Connectionism: Where Do We Go From Here?* PhD thesis, University of Alberta, 1998. 305iii, 305v, 305v, 305v

[8] J. C. Schlimmer. Concept acquisition through representational adjustment. Technical Report 87-19, Department of Information and Computer Science, University of California Irvine, 1987. 305ii

[9] S. B. Thrun and others. The MONK'S problems: A performance comparison of different learning algorithms. Technical Report CMU-CS-91-197, Carnegie Mellon University, December 1991. 305ii

[10] A. Vahed and C. W. Omlin. Rule extraction from recurrent neural networks using a symbolic machine learning algorithm. In *6th International Conference on Neural Information Processing*, page submitted, Perth, Australia, 1999. 305i