Using Extra Output Learning to Insert a Symbolic Theory into a Connectionist Network

M.R.W. DAWSON[1], D.A. MEDLER[2], D.B. MCCAUGHAN[3], L. WILLSON[4] and M. CARBONARO[5]

[1]*Department of Psychology, University of Alberta, Edmonton, Alberta, Canada T6G 2E9 (E-mail: mike@bcp.psych.ualberta.ca);* [2]*Center for the Neural Basis of Cognition, Carnegie Mellon University, 115 Mellon Institute, 4400 Fifth Avenue, Pittsburgh, PA 15213 USA;* [3]*Department of Computing Science, University of Alberta, Edmonton, Alberta, Canada T6G 2E9;* [4]*Department of Psychology, University of Alberta, Edmonton, Alberta, Canada T6G 2E9;* [5]*Department of Educational Psychology, University of Alberta, Edmonton, Alberta, Canada T6G 2E9*

Abstract. This paper examines whether a classical model could be translated into a PDP network using a standard connectionist training technique called extra output learning. In Study 1, standard machine learning techniques were used to create a decision tree that could be used to classify 8124 different mushrooms as being edible or poisonous on the basis of 21 different Features (Schlimmer, 1987). In Study 2, extra output learning was used to insert this decision tree into a PDP network being trained on the identical problem. An interpretation of the trained network revealed a perfect mapping from its internal structure to the decision tree, representing a precise translation of the classical theory to the connectionist model. In Study 3, a second network was trained on the mushroom problem without using extra output learning. An interpretation of this second network revealed a different algorithm for solving the mushroom problem, demonstrating that the Study 2 network was indeed a proper theory translation.

Key words: cognitive science, connectionist theories, symbolic theories

1. Introduction

One current debate in cognitive science centers on whether human information processing is best described in terms of a classical or a connectionist architecture (Bechtel and Abrahamsen, 1991; Born, 1987; Churchland, 1995; Churchland and Sejnowski, 1992; Clark, 1989; Clark, 1993; Dawson, 1991; Dawson, 1998; Dawson, Medler and Berkeley, 1997; Dawson and Schoptiocher, 1992a; Dawson and Shamanski, 1994; Fodor and McLaughlin, 1990; Fodor and Pylyshyn, 1988; Garson, 1994; Graubard, 1988; Horgan and Tienson, 1996; McCloskey, 1991; Pinker and Prince, 1988; Ramsey, Stich and Rumelhart, 1991; Schneider, 1987; Seidenberg, 1993; Smolensky, 1988; VanLehn, 1991). Much of this debate is rooted in the belief that connectionist networks and classical models are fundamentally different. For example, it is widely held that classical systems use explicit rules arranged in a hierarchy to manipulate symbols in a serial manner, whereas connectionist systems rely on parallel processing of sub-symbols via statistical procedures.

Minds and Machines **10:** 171–201, 2000.

To what extent does connectionism represent an alternative to the classical approach? There is a growing body of theoretical work that suggests that though these two views of cognitive science exhibit interesting differences, they are in fact highly similar (for example, see many of the papers in (Ramsey et al., 1991). These similarities are rooted in the fact that connectionists have *not* abandoned the foundational assumption that cognition is information processing. For instance, after reviewing a number of differences between digital computers and brains (Churchland, Koch and Sejnowski, 1990, p. 48, their italics) note that "these dissimilarities do not imply that brains are not computers, but only that *brains are not serial computers*." Later, after discussing some foundational issues concerning computation, they go on to point out "the question now is whether it is appropriate to describe various structures in nervous systems as computing. The summary answer is that it certainly is" (p. 50).

In short, while classical and connectionist researchers disagree about the specific details, they do agree on the general principle that cognition is information processing (see also Von Eckardt, 1993, pp. 125–141). Dawson (1998) has argued that this agreement means that the architectural debate between classical models and PDP networks must be carefully evaluated in the context of the very different kinds of analyses that must be performed to provide an account of information processing. Specifically, Dawson endorses the tri-level hypothesis proposed by such researchers (e.g., Marr, 1982; Pylyshyn, 1984), and argues that insight into this architectural debate will only be achieved if classical and connectionist theories are compared at the computational, algorithmic, and implementational levels of analysis.

1.0.1. *Computational level comparison*

At the computational level, we ask the question "What information processing problem is being solved by a system of interest" With respect to comparing the two views of cognitive science at this level, some classical researchers have argued that the connectionist architecture does not have the computational power to solve the same kind of problems as the classical architecture (e.g., Lachter and Bever, 1988). At the same time, some connectionist researchers (McClelland, 1992) have argued that classical approaches do not actually possess the power that they claim (especially with regards to the compositionality of language) and that it is the connectionist framework that captures the true nature of language.

However, it has been long established that connectionist networks have the same in principle computational power as do classical architectures. In some of the earliest work on neural networks, McCulloch and Pitts (1943) examined finite networks whose components could perform simple logical operations like AND, OR, and NOT. They were able to prove that such systems could compute any function that required a finite number of these operations. From this perspective, the network was only a finite state automaton (see also Hopcroft and Ullman, 1979; Minsky, 1972). However, McCulloch and Pitts went on to show that a Universal Turing

Machine (UTM) could be constructed from such a network, by providing the network the means to move along, sense, and rewrite an external “tape” or memory. “To psychology, however defined, specification of the net would contribute all that could be achieved in that field” (McCulloch and Pitts, 1943/1988, p. 25).

Some more modern results have established the equivalence between classical and connectionist architectures with respect to specific computational and representational issues. For example, Fodor and Pylyshyn (1988) have pointed out that thought is highly structured, and as a result it is very systematic. This means that if an information processor can express one belief (e.g., “John loves Mary”), then it should also be capable of expressing related beliefs built from the same components (e.g., “Mary loves John”). However, Fodor and Pylyshyn argue that connectionist representations are not structured in this way, and as a result the principles of connectionism cannot, by themselves, explain the systematicity of thought (pp. 48–50). In response to this argument, Hadley (Hadley, 1994a, 1994b; Hadley, 1997) has argued that Fodor and Pylyshyn’s (1988) notion of systematicity glosses over several important distinctions, and has proposed a more sophisticated notion in which systematicity can be exhibited to different degrees. This reformulation is strongly tied to the ability of a system to generalize its behavior to new stimuli, and defines degrees of systematicity in terms of varying degrees of generalization. Hadley has shown that many different connectionist networks achieve sufficiently high degrees of sytematicity. In addition, Hadley and Hayward (1997) have described a network that demonstrates Hadley’s strongest degree of systematicity.

Other modern results have validated and extended the pioneering research of McCulloch and Pitts (1943/1948). For instance, one popular kind of PDP model is a recurrent network, which can process temporal stimuli because some of its components act as an internal memory. Recurrent networks are computationally very powerful. Kremer (1995) has shown that a particular type of recurrent network (Elman, 1990) is formally equivalent to a discrete finite automaton. Given the existence of this type of equivalence, it is not surprising that recurrent networks can be used to construct the machine head of a UTM (Williams and Zipser, 1989). However, more interesting results have involved determining whether all of the components of a UTM could be constructed within a single network (e.g., Siegelmann, 1999). Early work of Siegelman and Sontag (1991) developed a proof that a such a network was possible in principle, but this proof limited the absolute size of this network to a relatively large value (a maximum of 10^5 processing units). Kilian and Siegelmann (1993) have developed a general proof that recurrent networks that use logistic activation functions are indeed equivalent to Turing machines. One specific example of such equivalence has been provided by Siegelmann and Sontag (1995), who proved that Minsky’s (1972) well known 4-symbol, 7-machine state UTM could be built from a recurrent network that used 886 processing units. “Turing universality is a relatively common property of recurrent neural network modes” (Kilian and Siegelmaun, 1993, p. 137).

These results show that classical models and PDP networks are, in principle, computationally equivalent. This is because, for either architecture, one can defend the claim that they have the same competence as a UTM. This kind of claim is important for two reasons. First, it establishes one kind of identity between the two different approaches to cognitive science. Second, given this relationship at the computational level, it now makes sense to compare classical and connectionist models in terms of the algorithms that they carry out.

1.0.2. *Algorithmic level comparison*

At the algorithmic level, we ask the question “What specific information processing steps are being carried out to solve an information processing problem?”. Given the computational equivalence of classical and connectionist architectures, it is now important for cognitive scientists to determine (a) whether the two architectures carry out qualitatively different algorithms, and (b) if the algorithms are different, then which provides a better account of human cognition (Dawson, 1998).

Why, at the algorithmic level, are PDP networks thought to be different from classical theories? One reason is network “appearance” – at first glance, PDP networks do not look like classical algorithms (Churchland and Sejnowski, 1989). A second reason – which increases reliance on network appearance – is that the internal structure of a trained network is extremely difficult to interpret (Andrews, Diederich, and Tickle, 1995; Hecht-Nielsen, 1987; McCloskey, 1991; Mozer and Smolensky, 1989; Seidenberg, 1993; Smith, 1996, pp 64–65). As a result, detailed algorithmic accounts of how a PDP network converts its inputs into an output response are rarely seen in the literature, and are even less frequently compared to classical algorithms. Marvin Minsky has pointed out that “connectionists take pride in not understanding how a network solves a problem” (Stork, 1997, p. 18).

In recent years, however, some researchers have developed techniques for investigating the internal structure of PDP networks (Andrews et al., 1995; Berkeley, Dawson, Medler, Schopflocher, and Homsby, 1995; Gallant, 1993; Hanson and Burr, 1990; Hinton, 1986; McCaughan, 1997; Omlin and Giles, 1996). In some cases the application of these techniques has revealed that a network’s algorithm can be much more similar to a classical theory than one might initially expect (Dawson et al., 1997). For example, Berkeley et al. (1995) analyzed a network that had been trained on a logic task, and discovered in its internal structure five different network states that corresponded to traditional rules of logic. They argued that this result blurred the difference between classical and connectionist accounts of cognition.

The studies in the current paper represent an attempt to go beyond a mere “blurring” of the differences between a classical algorithm and a connectionist algorithm. In the philosophy of science, if two apparently different theories are in fact identical, then one should be able to translate one theory into the other. This is called *intertheoretic reduction* (Churchland, 1985; Churchland, 1988; Hooker, 1979; Hooker, 1981). The widely accepted view that classical and connectionist cognitive science are fundamentally different (Schneider, 1987) amounts to the claim that intertheoretic reduction between a symbolic model and a PDP network is impossible. Below, we examine this belief directly by asking whether we can translate a classical algorithm into a PDP network using standard connectionist training techniques.

1.1. EXTRA OUTPUT LEARNING AND ALGORITHM INSERTION

One of the most crucial stages in the design of a pattern recognition system is selecting the correct set of input features (Ripley, 1996). This can be thought of as a stage in which the input patterns are preprocessed. Such preprocessing often involves identifying those features in the input that are relevant for classification and those features that are redundant and therefore can be removed. In other words, preprocessing is typically used to reduce the amount of information at the input, by eliminating information that is not thought to be useful for the task at hand.

However, another perspective on preprocessing the data would be to add information to aid the pattern recognition task. A pattern classification system is normally only informed about what the correct label for a pattern should be. For instance, later in this paper we describe a mushroom classification problem, in which a system would normally only be taught to generate the label "edible" or the label "poisonous" when presented a set of mushroom features. But, it is often the case that more information than this is actually available. Specifically, there often exists prior information about *why* an input pattern belongs to one class or another. This information could be included either with the inputs or with outputs. Adding this information to the outputs, however, had one distinct advantage: this structure (i.e., the extra output units and any connections feeding into them) can be removed from the network following training without affecting the network's performance on the primary task.

Thus, one could add this information to the pattern classification problem by teaching the system not only to generate a label of interest (e.g., "edible", "poisonous") but to also generate a reason for assigning this label (e.g., "passed Rule 1", "failed Rule 4"). Adding this information amounts to requiring the system to make a more complex categorization of the instances that it is presented. This is because the network in essence has to assign each instance to a major category of interest as well as to a subcategory which represents the reason for making this assignment.

Elaborating a classification task along the lines described above has been called *injection of hints or extra output learning* (Abu-Mostafa, 1990; Caruana and de Sa, 1997; Gallmo and Carlstrom, 1995; Suddarth and Kergosien, 1990; Suddarth, Sutton, and Holden, 1988; Yu and Simmons, 1990). It has been found that extra output learning often speeds up the training of PDP networks because the requirement to subcategorize input patterns places constraints on the potential configurations of network weights. This helps restrict the "search space" that is traversed during training as the network attempts to find a state which minimizes its overall error.

We were not interested in using extra output learning to affect the speed of training a PDP network. Instead, we hypothesized that extra output learning could be used as a technique to insert a classical algorithm into a PDP network while the network was being trained to solve a pattern recognition problem. Imagine a identification problem in which input features are used to classify mushrooms as being edible or poisonous. Imagine further a set of classical rules that are known to accomplish this task (e.g., Rule x, Rule y, Rule z). With the existence of these rules, one could use extra output learning to train a network to generate responses of the type “this mushroom is poisonous because of Rule z” for every mushroom in the training set. If the network could learn to make these types of assertions, then it would stand to reason that the network had internalized the classical algorithm during training. Therefore, we should be able to interpret the internal structure of the trained network, and discover a precise mapping between network states and the rules that define the classical algorithm. The studies below test this hypothesis.

The remainder of this paper proceeds as follows: First, we describe a particular pattern classification problem, and use traditional machine learning techniques to derive a classical algorithm capable of solving it. Second, we use extra output learning in an attempt to insert this classical algorithm directly into a PDP network. An analysis of this network is presented to show that a precise mapping was achieved between internal network states and the rules of the classical algorithm. Third, we discuss the training of a second network, in which no attempt is made to insert the classical rules, and we show that this network does not map onto the classical algorithm. This demonstrates that our intertheoretic reduction is genuine; it is not an artifact of the particular classification problem that we selected.

2. Study 1

2.1. METHOD

The field of machine learning is concerned with the development of automatic procedures that are capable of learning to correctly classify a set of example patterns (Ripley, 1996). Each pattern is a set of features. Consequently, one task of a machine learning algorithm is to discover how to use some or all of these features to determine a label or class for each pattern.

Importantly, this is not the only task of a machine learning algorithm. In addition, it is required to generate a mapping between features and labels that can be comprehended and used by humans (Michie, Speigelhalter, and Taylor, 1994). “Machine Learning aims to generate classifying expressions simple enough to be understood easily by humans. They must mimic human reasoning sufficiently well to provide insight into the decision process” (p. 2). For this reason, many machine learning algorithms are classical in nature, and deliver a set of rules or a program which, if followed, will tell a human exactly how to use observed features to classify patterns.

One example of a classical machine learning algorithm is a technique that will induce a decision tree from a set of examples, such as the ID3 procedure (Quinlan, 1986). A decision tree is a hierarchically structured classifier. The tree starts at a root node, and branches outward from this node into intermediate nodes and, eventually, into a set of terminal leaves. Each terminal leaf represents a label that is assigned to a pattern. In the 1D3 algorithm, each node in the tree represents a decision used to "split" training examples into positive or negative instances. In other words, the outcome of a test at a node determines where to send the pattern for its next evaluation in the tree. This is continued until the pattern is assigned a label by being moved into a terminal leaf. Consequently, any decision tree produced by the ID3 algorithm is equivalent to a series of classical inference rules (e.g., *married(m)* Λ *man(m)* → *bachelor (m))*. The set of decision rules can easily be understood by human users of an ID3 program. We used a variation of the 1D3 algorithm to induce a decision tree for a benchmark problem in the machine learning literature. We wanted this problem to be a real world problem that would be challenging to an artificial (or human) classifying system, so that the resulting classical theory would be rich and nontrivial.

The problem that we selected was the classification of mushrooms as being either edible or poisonous on the basis of a number of observable features (Schlimmer, 1987). The data set consisted of the hypothetical description of 23 different mushrooms in the *Agaricus* and *Lepiota* family (Lincoff, 1981. pp. 500–525). Each mushroom was described as a set of the 21 different features that are shown in Table 1. Multiple featural descriptions of one species of mushroom are possible because one species might be found in several different habitats, have more than one possible odor, etc. As a result, the total data set consisted of 8124 different instances. 4208 of these patterns corresponded to edible mushrooms; the remaining 3916 patterns corresponded to inedible mushrooms (i.e., mushrooms that were definitely poisonous, or were of unknown edibility and therefore not recommended). The mushroom database can be retrieved via ftp from ftp.ics.uci.edu: pub/machine-learning-databases, or through the WWW at http://www.ics.uci.edu>mlearn /MLRepository.html.

2.2. INDUCING A DECISION TREE FOR THE MUSHROOM PROBLEM

A variation of the 1D3 algorithm (Quinlan, 1986) that was developed by one of the authors (Medler, 1998) was used to induce a decision tree for the mushroom problem. Whereas the original 1D3 algorithm is limited to creating trees with at most two children at each node (e.g., *yellow,* ∼*yellow*), the new algorithm was modifed to allow multiple branching from each node (e.g., *yellow, white, purple, orange,* etc.). The decision tree that was generated by the algorithm defined a sequence of 5 rules that could be used to correctly classify all 8124 of the example mushrooms. This set of rules is given in Table 2. An examination of this table indicates clearly that these rules are classical in nature. They are explicit, local,

Table I. The 21 different types of features, and their possible values, used in the Schlimmer (1987) mushroom classification problem

Mushroom Feature	Possible Values of the Feature
Cap Shape	Bell, conical, convex, flat, knobbed, sunken
Cap Surface	Fibrous, grooves, scaly, smooth
Cap Color	Brown, buff, cinnamon, gray, green, pink, purple, red, white, yellow
Bruises	No bruises, bruises
Odor	Almond, anise creosote, fishy, foul, musty, none, pungent, spicy
Gill Attachment	Attached, descending, notched
Gill Spacing	Close, crowded, distant
Gill Size	Broad, narrow
Gill Color	Black, brown, buff, chocolate, gray, green, orange, pink, purple, red, white, yellow
Stalk Shape	Enlarging, tapering
Stalk Surface Above Ring	Fibrous, scaly, silky, smooth
Stalk Surface Below Ring	Fibrous, scaly, silky, smooth
Stalk Color Above Ring	Brown, buff, cinnamon, gray, orange, pink, red, white, yellow
Stalk Color Below Ring	Brown, buff, cinnamon, gray, orange, pink, red, white, yellow
Veil Type	Partial, universal
Veil Color	Brown, orange, white, yellow
Ring Number	None, one, two
Ring Type	Cobwebby, evanescent, flaring, large, none, pendant, sheathing, zone
Spore Print Color	Black, brown, buff, chocolate, green, orange, purple, white, yellow
Population	Abundant, clustered, numerous, scattered, several, solitary
Habitat	Grasses, leaves, meadows, paths, urban, wastes, woods

and digital (Haugeland, [985). Furthermore, executing this algorithm would require that each rule be considered in a particular serial order.

3. Study 2

The results of Study 1 have provided us with a set of five tests or rules that represent a classical algorithm for solving the mushroom problem. The purpose of Study 2 was to see whether this algorithm could be inserted into a PDP network using extra output learning. This required us to train a network using extra output learning, and then to analyze its internal structure to determine whether there existed a mapping from network states to the rules that defined the classical algorithm.

Table II. A sequence of five rules that define a decision tree for the mushroom problem, and which can be used to correctly classify each of the 8124 instances as being edible or not. The bold labels "Rule 1 Edible", "Rule 1 Poisonous", etc. indicate the 9 decision points in the algorithm at which a mushroom will be classified as being edible or not. These decision points are used in Study 2 to define extra outputs when an attempt was made to translate this algorithm into a PDP network

Step 1	*What is the mushroom's odor?*
	If it is almond or anise then it is edible. **(Rule 1 Edible**)
	If it is creosote or fishy or foul or musty or pungent or spicy then it is poisonous. **(Rule 1 Poisonous**)
	If it has no odor then proceed to Step 2.
Step 2	*Obtain the spore print of the mushroom.*
	If the spore print is black or brown or buff or chocolate or orange or yellow then it is edible. (**Rule 2 Edible**)
	If the spore print is green or purple then it is poisonous. **(Rule 2 Poisonous)**
	If the spore print is white then proceed to Step 3.
Step 3	*Examine the gill size of the mushroom.*
	If the gill size is broad, then it is edible. **(Rule 3 Edible)**
	If the gill size is narrow, then proceed to Step 4.
Step 4	*Examine the stalk surface above the mushrooms ring.*
	If the surface is fibrous then it is edible. **(Rule 4 Edible)**
	If the surface is silky or scaly then it is poisonous. **(Rule 4 Poisonous)**
	If the surface is smooth the proceed to Step 5.
Step 5	*Examine the mushroom for bruises.*
	If it has no bruises then it is edible. **(Rule 5 Edible)**
	If it has bruises then it is poisonous. **(Rule 5 Poisonous)**

3.1. METHOD

Input Units. The network that was trained had 21 different input units, one for each mushroom feature in the data set. All of the features were coded as discrete activation values between 0.0 and 1.0. Each activation value corresponded to a different value of the particular feature encoded in that input unit. For example, if there were four different values of a feature, they would be represented by setting the input value of that feature's unit to 0.0, 0.33, 0.66, or 1.0, depending on which of the four values of that feature were to be presented to the network at that time.

Hidden Units. The network was trained with five hidden units, because pilot tests indicated that if fewer hidden units were used, then the network would fail to find a solution to the problem. All of the hidden processors were *value units* (Dawson and Schopflocher, 1992b). A value unit is similar to the processing units found in standard multilayer perceptrons trained using error backpropagation (Rumelhart, Hinton and Williams, 1986). However, instead of using a sigmoid activation function (such as the logistic equation), value units use a Gaussian activation function that has a minimum of 0, a maximum of 1, and a standard deviation of 1. As a result, a value unit will only generate high activity to a narrow range of incoming signals. We elected to use the value unit architecture in the current study because previous results have shown that it permits us to train networks with fewer hidden units, that networks can be trained more quickly, and that the trained network is likely to be easier to interpret than is the case with more traditional types of processing units (Berkeley et al., 1995; Dawson, 1990; Dawson et al., 1997; Dawson and Schopflocher, 1992b; Dawson, Shamanski, and Medler, 1993).

Output Units. Ten different output value units were used in the network. One output unit encoded the edible/poisonous classification, and the other 9 output units were used to inject the hints that were available from the classical algorithm that we had obtained in Study 1. Table 3 provides the mapping between the classical algorithm and the network's output units. As can be seen from the table, the extra output units were used to encode the decision point in the algorithm at which each mushroom was classified as being edible or poisonous. There were 9 additional output units because, as can be seen from Table 2, there were only 9 different decision points in the algorithm. In other words, for each mushroom pattern, the network of value units was trained to activate two output units. One of these units indicated whether the mushroom was poisonous or not. The other unit indicated the point in the classical algorithm could be used to justify the network's classification. The complete network structure is illustrated in Figure 1.

Training the Network. The complete network, which is illustrated in Figure 1. was trained using the variation of the generalized delta rule designed for value unit networks (Dawson and Schopflocher, 1992b), using a learning rate of 0.005 and no momentum. Prior to training, the network's connection weights were randomized

Table III. The translation from the decision point in the classical algorithm (from Table 3) to a 10-bit output vector for the network. In the output vector, the first bit indicates whether the presented mushroom is edible (1) or not (0). One of the other bits is turned on to indicate the decision point in the classical algorithm at which the edible/not edible decision would have been made

Decision Point in the Algorithm	Extra Output Encoding for the Network
Rule 1 Edible	1 1 0 0 0 0 0 0 0 0
Rule 1 Poisonous	0 0 1 0 0 0 0 0 0 0
Rule 2 Edible	1 0 0 1 0 0 0 0 0 0
Rule 2 Poisonous	0 0 0 0 1 0 0 0 0 0
Rule 3 Edible	1 0 0 0 0 1 0 0 0 0
Rule 4 Edible	1 0 0 0 0 0 1 0 0 0
Rule 4 Poisonous	0 0 0 0 0 0 0 1 0 0
Rule 5 Edible	1 0 0 0 0 0 0 0 1 0
Rule 5 Poisonous	0 0 0 0 0 0 0 0 0 1

between ± 1.0, and the biases of all value units (i.e., the mean of the Gaussian for each unit) were initialized to zero. Network connections and biases were updated after every pattern presentation, and pattern presentation was randomized every epoch. (In our procedure, one epoch consists of one presentation of each of the 8124 patterns in the training set.) The network was trained until a ‘hit’ was recorded for every output unit for every pattern in the training set. A hit was defined as being an activation of 0.95 or greater when the desired output was 1, and as being an activation of 0.05 or less when the desired output was 0. Convergence (i.e., a hit on every output unit for every pattern) was achieved after 8699 epochs of training.

4. Results

The purpose of Study 2 was to insert the classical algorithm that was obtained in Study 1 into a PDP network. The fact that the network converged to a solution to the extra output learning version of the mushroom problem does not by itself indicate that this translation was successful. To determine whether the classical algorithm was actually converted into a network requires an interpretation of the

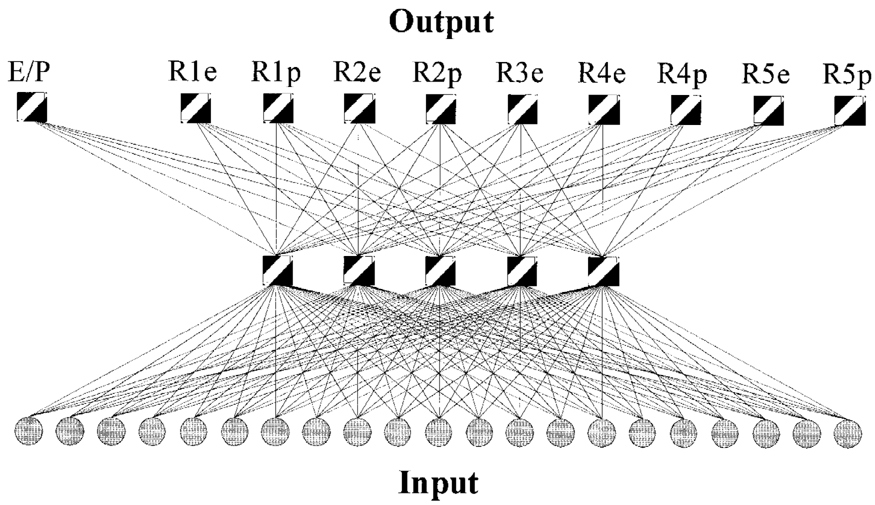

Figure 1. The neural network that was trained in Study 2. All of the output and all of the hidden processors were value units, as indicated by the dark squares with the white diagonal ber running through them. The leftmost output unit, labeled "E/P" was trained to represent the primary classification (i.e., whether a presented mushroom was edible or poisonus). The remaining 9 output units provided the extra output learning as described in the text, where "R1e" stands for "Rule 1 edible", "R1p" stands for "Rule one poisonous", and so on.

internal states of the PDP model (i.e., patterns of hidden unit activations), and a mapping of these internal states back to the classical algorithm.

Cluster Analysis of Hidden Unit Activities. In previous research (Berkeley et al., 1995; Dawson et al., 1997), the internal structure of value unit networks was interpreted by exploiting local regularities within each hidden unit. However, this technique will not work in all cases. This is because it is blind to regularities that can be represented as patterns of activities distributed across sets of hidden units. We have found that cluster analysis of hidden unit activities is an alternative approach to network interpretation that overcomes this problem, and which provides rich interpretations of the internal structure of trained networks.

The first step in a cluster analysis of a network of value units is to “wiretap” the hidden units by recording the activity of each hidden processor when each of the training patterns are presented to the network after it has converged on a solution to the desired problem. (This wiretapping phase was also central to the interpretation of the local structure of units in our previous research (Berkeley et al., 1995; Dawson et al., 1997).) For example, after wiretapping the network trained in Study 2, we created a data matrix consisting of 5 columns (one for each hidden unit) and 8124 rows (one for each stimulus in the training set). Each entry in this data matrix represented the activations produced in a hidden unit when one of the mushrooms was presented to the network.

The second step is to perform a k-means clustering of the data matrix created by wiretapping the network. The k-means algorithm is an iterative procedure that assigns data points to k different clusters in such a way that each member of a cluster is closer to the centroid of that cluster than to the centroid of any other cluster to which other data points have been assigned. Whenever cluster analysis is performed, one question that must be answered is "How many clusters should be used?" (in other words, what should the value of k be?). Unfortunately, no single method for determining the optimal number of clusters in a data set has been agreed upon (Aldenderfer and Blashfield, 1984; Everitt, 1980). This is reflected in the fact that a large number of different types of methods exist for dealing with this issue (Milligan and Cooper, 1985).

While no general method exists for determining the optimal number of clusters, one can take advantage of heuristic information concerning the domain that is being clustered to come up with a satisfactory method for this domain. When the hidden unit activities of a trained network are being clustered, we know that there is a correct mapping from these activities to output responses. This is because if the network has correctly learned the task that it was presented, then the network itself has discovered one such mapping. This knowledge can be used to create the following heuristic stopping rule: extract the smallest number of clusters such that every hidden unit activity vector in the cluster produces the same output response in the network. We found that when the hidden unit activities were assigned to 12 different clusters that each cluster mapped onto a unique network output, indicating that this was the appropriate number of clusters to use to describe this network. Table 4 illustrates the mapping from these 12 clusters to the 9 different outputs that the network was trained to generate.

Cluster Interpretation. Once the k-means analysis of the "wiretap" data has been completed, the third step in network analysis is to identify the input features shared by all of the members of each cluster. This is done using the same technique that was used to identify "definite features" associated with local structures found in individual hidden value units (Berkeley et al., 1995). For each cluster, we compute the mean and standard deviation of each of the 21 input features, looking for features that are constant across all cluster members (i.e., features with a standard deviation of 0). When this analysis was performed for the 12 clusters extracted from the Study 2 network, a large number of such definite features were identified in each cluster. These definite features are detailed in Table 5.

From Clusters to Rules. The sets of definite cluster features listed in Table 5 can be thought of as conditions used by the network to judge whether a mushroom is edible or not. For instance, one of the "network rules" for identifying a mushroom as being poisonous is the conjunction of all of the features listed for Cluster 1. However, while "network rules" of this type are perfectly legitimate mushroom classifiers, they are more complicated than is necessary. Some of the features in

Table IV. Crosstabulation table indicating the frequency of patterns classified in terms of a) the cluster to which they belong and b) the network output that they produce

Cluster	Output States of the Network								
	State 1 0010000000 (R1P)	State 2 1100000000 (R1E)	State 3 1001000000 (R2E)	State 4 0000100000 (R2P)	State 5 1000010000 (R3E)	State 6 1000000010 (R5E)	State 7 0000000100 (R4P)	State 8 0000000001 (R5P)	State 9 1000001000 (R4E)
1	3796	0	0	0	0	0	0	0	0
2	0	704	0	0	0	0	0	0	0
3	0	96	0	0	0	0	0	0	0
4	0	0	0	0	528	0	0	0	0
5	0	0	0	0	0	0	40	0	0
6	0	0	0	72	0	0	0	0	0
7	0	0	0	0	0	0	0	0	12
8	0	0	0	0	0	12	0	0	0
9	0	0	2832	0	0	0	0	0	0
10	0	0	0	0	0	0	0	8	0
11	0	0	0	0	0	0	0	0	12
12	0	0	0	0	0	12	0	0	0

Table V. Definite features for each of the clusters of hidden unit activities from the Study 2 network. The feature labeled “N” indicates the number of patterns in the cluster

Feature	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6
N	3796	704	96	528	40	72
Cap Shape	~ Bell ~ Conical ~ Sunken	~ Conical ~ Knobbed ~ Sunken	Convex or Flat	~ Conical ~ Sunken	~ Sunken Flat	Bell or
Cap Surface	~ Grooves	~ Fibrous ~ Grooves	~ Grooves ~ Scaly	~ Grooves	~ Grooves ~ Smooth	~ Fibrous ~ Grooves
Cap Color	~ Green ~ Purple	Brown or White or Yellow	White or Yellow	~ Green ~ Purple ~ Yellow	Brown or Yellow	Buff or Pink or White
Bruises		Bruises	Bruises		No Bruises	Bruises
Odor	Creosote or Fishy or Foul or Musty or Pungent or Spicy	Almond or Anise	Almond or Anise	None	None	None
Gill Attachment	~ Descending ~ Notched	Free	Free	Free	Free	Free
Gill Spacing	~ Distant	Close	Crowded	Close or Crowded	Close or Crowded	Close
Gill Size		Broad	Narrow	Broad	Narrow	Broad
Gill Color	~ Green ~ Orange ~ Red	Black or Brown or Gray or Pink or White	Brown or Pink or White	Gray or Pink or Red or White	White or Yellow	Gray or Green or White
Stalk Shape		Enlarging	Tapering	Enlarging	Enlarging	Enlarging
Stalk Surface Above Ring	~ Scaly	Smooth	Smooth	~ Fibrous	Scaly or Silky	Smooth
Stalk Surface Below Ring		Silky or Smooth	Smooth	~ Fibrous	Scaly	Smooth
Stalk Color Above Ring	~ Gray ~ Orange ~ Red ~ Yellow	White	White	Brown or Red or White	White or Yellow	White
Stalk Color Below Ring	~ Gray ~ Orange ~ Red ~ Yellow	White	White	Brown or Red or White	Brown or White or Yellow	White

Table V. Continued

Feature	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6
Veil Type	Partial	Partial	Partial	Partial	Partial	Partial
Veil Color	White	White	White	White	White or Yellow	White
Ring Number	~ Two	One	One	Two	One	Two
Ring Type	~ Cobwebby ~ Flaring ~ Sheathing ~ Zone	Pendant	Pendant	Evanescent or Pendant	Evanescent	Pendant
Ring Number	~ Two	One	One	Two	One	Two
Ring Type	~ Cobwebby ~ Flaring ~ Sheathing ~ Zone	Pendant	Pendant	Evanescent or Pendant	Evanescent	Pendant
Spore Print Color	~ Buff ~ Green ~ Orange ~ Purple ~ Yellow	Black or Brown	Brown or Purple	White	White	Green
Population	~ Abundant ~ Numerous	Numerous or Scattered or Solitary	Several	~ Abundant	Clustered Or Several	Several
Habitat	~ Meadows ~ Waste	Grasses or Meadows or Paths	Woods	~ Leaves ~ Meadows ~ Urban	Leaves or Woods	Grasses or Meadows

Feature	Cluster 7	Cluster 8	Cluster 9	Cluster 10	Cluster 11	Cluster 12
N	12	12	2832	8	12	12
Cap Shape	~ Bell ~ Conical ~ Sunken	~ Bell ~ Conical ~ Sunken	~ Conical	~ Convex ~ Sunken	~ Bell ~ Conical ~ Sunken	~ Bell ~ Conical ~ Sunken
Cap Surface	~ Grooves ~ Smooth	~ Grooves ~ Smooth	~ Grooves	~ Fibrous ~ Smooth	~ Grooves ~ Smooth	~ Grooves ~ Smooth
Cap Color	Brown or Cinnamon	Brown or Cinnamon	~ Buff ~ Cinnamon ~ Pink ~ Yellow	White	Brown or Cinnamon	Brown or Cinnamon
Bruises	No Bruises	No Bruises	No Bruises	Bruises	No Bruises	No Bruises
Odor	None	None	None	None	None	None
Gill Attachment	Free	Free	Attached or Free	Free	Free	Free

Table V. Continued

Feature	Cluster 7	Cluster 8	Cluster 9	Cluster 10	Cluster 11	Cluster 12
N	12	12	2832	8	12	12
Gill Spacing	Crowded	Crowded	Close or Crowded	Crowded	Crowded	Crowded
Gill Size	Narrow	Narrow		Narrow	Narrow	Narrow
Gill Color	White	White	∼ Buff ∼ Green ∼ Red	White	White	White
Stalk shape	Enlarging	Enlarging		Enlarging	Enlarging	Enlarging
Stalk Surface Above Ring	Fibrous	Fibrous	∼ Scaly ∼ Silky	Smooth	Fibrous	Smooth
Stalk Surface Below Ring	Fibrous	Fibrous	∼ Scaly ∼ Silky	Smooth	Smooth	Smooth
Stalk Color Above Ring	White	White	Gray or Orange or Pink or White	White	White	White
Stalk Color Below Ring	Brown	Brown	Gray or Orange or Pink or White	White	Brown	Brown
Veil Type	Partial	Partial	Partial	Partial	Partial	Partial
Veil Color	White	White	∼ Yellow	White	White	White
Ring Number	One	One	One	One	One	One
Ring Type	Evanescent	Evanescent	Evanescent or Flaring or Pendant	Pendant	Evanescent	Evanescent
Spore Print Color	White	White	Black or Brown or Buff or Chocolate or Orange or Yellow	White	White	White
Population	Several	Several	∼ Numerous	Clustered	Several	Several
Habitat	Leaves	Leaves	∼ Meadows ∼ Paths ∼ Waste	Leaves	Leaves	Leaves

the rule given above are not diagnostic, and therefore could be deleted from the rule. For example, “partial veil” is one of the definite features for Cluster 1. However, this feature is true for all of the mushrooms in the training set, and therefore is irrelevant to the task of making a judgement about a mushroom. It would be extremely useful if unnecessary features like “partial veil” could be removed from the description of the clusters in order to simplify our account of how the network maps inputs to outputs.

One general approach for simplifying the featural accounts of clusters is to perform a linear discriminant analysis of their features (see Study 3 below). However, for the network in Study 2, a more specific approach is available. As this network is an attempt to translate a set of classical rules into a PDP model, it should be possible to simplify the description of the network’s algorithm by finding a mapping from the clusters to the classical algorithm.

Our first step in seeking this mapping was to translate the classical algorithm into an intermediate form that could tractably be mapped onto network states. As was discussed earlier, this kind of translation is central to new wave reductionism. Our translation consisted in converting the step-by-step program given in Table 1 into an equivalent set of production rules. These rules simply describe the properties of mushrooms that must be true at each decision point. For instance, at the “Rule 1 Edible” decision point, one could create the production rule “If the odour is anise or almond, then the mushroom is edible”. Similar productions can be created for later decision points in the algorithm, but these productions will involve a longer list of mushroom features. The complete set of productions that were created for the decision tree algorithm are provided in Table 6.

The next step in the new wave reduction of the classical algorithm to the PDP network is to establish a mapping between the Table 6 productions and network states. First, note that with the encoding used for the extra output learning, each production in Table 6 is represented by a unique output unit encoding for the network (see Table 2). This is because each production is basically an elaborate account of each decision point in the decision tree. Second, note that we have already established that each cluster of hidden unit activities maps uniquely onto an output unit encoding (see Table 4). In other words, there exists a unique mapping from internal network states (i.e., vectors of hidden unit activities) to the productions that define a classical algorithm. The complete mapping from hidden unit clusters to the productions is given in Table 6.

5. Discussion

The analysis of the Study 2 network has revealed a new wave intertheoretic reduction between a classical algorithm and a PDP model. We found that a k-means cluster analysis of the mushroom network’s hidden unit activities produced 12 clusters, each of which was associated with a rich set of definite features. We also found a precise mapping between these clusters and a set of production rules that

Table VI. Translation of the decision points from the decision tree generated in Study 1 (see Table 2) into an equivalent set of 9 different production rules

Decision Point from Table 2	Corresponding Cluster		Equivalent Production
Rule 1 Edible	2 or 3	**P1:**	if (odor = anise) $\vee$ (odor = almond) $\rightarrow$ edible
Rule 1 Poisonous	1	**P2:**	if (odor $\neq$ anise) $\wedge$ (odor $\neq$ almond) $\wedge$ (odor $\neq$ none) $\rightarrow$ not edible
Rule 2 Edible	9	**P3:**	if (odor = none) $\wedge$ (spore print color $\neq$ green) $\wedge$ (spore print color $\neq$ purple) $\wedge$ (spore print color $\neq$ white) $\rightarrow$ edible
Rule 2 Poisonous	6	**P4:**	if (odor = none) $\wedge$ ((spore print color = green) $\vee$ (spore print color = purple)) $\rightarrow$ not edible
Rule 3 Edible	4	**P5:**	if (odor = none) $\wedge$ (spore print color = white) $\wedge$ (gill size = broad) $\rightarrow$ edible
Rule 4 Edible	7 or 11	**P6:**	if (odor = none) $\wedge$ (spore print color = white) $\wedge$ (gill size = narrow) $\wedge$ (stalk surface above ring = fibrous) $\rightarrow$ edible
Rule 4 Poisonous	5	**P7:**	if (odor = none) $\wedge$ (spore print color = white) $\wedge$ (gill size = narrow) $\wedge$ ((stalk surface above ring = silky) $\vee$ (stalk surface above ring = scaly)) $\rightarrow$ not edible
Rule 5 Edible	8 or 12	**P8:**	if (odor = none) $\wedge$ (spore print color = white) $\wedge$ (gill size = narrow) $\wedge$ (stalk surface above ring = smooth) $\wedge$ (bruises no) $\rightarrow$ edible
Rule 5 Poisonous	10	**P9:**	if (odor = none) $\wedge$ (spore print color = white) $\wedge$ (gill size = narrow) $\wedge$ (stalk surface above ring = smooth) $\wedge$ (bruises = yes) $\rightarrow$ not edible

represented the decision points in the decision tree algorithm that was discovered in Study 1. In turn, there is a direct mapping from any of the 9 productions back to the decision tree algorithm. This provides extremely strong evidence that we were able to use extra output learning to provide an exact translation of the classical algorithm into the network of value units.

However, one further point must be tested before considering the implications of this intertheoretic reduction. It is possible that the only way of categorizing the mushrooms was with the algorithm that was derived in Study 1. If this is the case, then the network would be forced to derive this algorithm when it converged on a solution to the problem, and our notion of theory translation would be brought into question. To increase our confidence in the view that the classical algorithm had indeed been translated into a PDP network, it is important to show that when extra output learning is not used then the network can solve the mushroom problem using a procedure that does *not* correspond to the classical algorithm.

6. Study 3

The purpose of this study was to determine whether a PDP network could discover a different algorithm (i.e., one that does not correspond to the Study 1 decision tree) to solve the mushroom problem. To accomplish this, we trained a different network of value units on the mushroom problem. In this case, we did not use extra output learning. We simply used one output unit to encode whether the presented mushroom was poisonous or not. After training the network, we used cluster analysis to interpret its internal structure. This analysis revealed that the network had discovered a different procedure for classifying the mushrooms, which in turn supports the claim that Study 2 represents a true translation of a classical theory into a PDP network.

6.1. METHOD

Training Set. The network was trained on the identical set of 21 input features, encoded using the same technique as described in Study 2. The only change in the training set was with respect to the output activation – the network was only required to judge whether a mushroom was edible or not. In other words, extra output learning was not used in this study.

Network Architecture. The network had 21 input units, four hidden value units, and one output value unit. This architecture was selected because pilot results demonstrated, in contrast to Study 2, that when extra output learning was not used a network with only four hidden units could learn this task. The output value unit was trained to generate a response of “1” to an edible mushroom, and a response of “0” to an inedible mushroom. Initial connection weights for the network were randomly selected from the range from –1.0 to 1.0. The biases of each hidden unit and of the output unit were set to 0, and were not modified during training.

Network Training. The network was trained with the Dawson and Schopflocher (1992) learning rule, with a learning rate of 0.01 and with no momentum. The network converged (i.e., achieved a hit on every pattern) after 1852 sweeps through the training set.

7. Results

Cluster Analysis of the Network. In order to interpret how this network was solving the mushroom problem, k-means cluster analysis was performed on the 8124 vectors of hidden unit activities that were obtained by “wiretapping” the four hidden units of this network after it had converged. Once again, we extracted the smallest number of clusters such that each member of each cluster mapped onto the same network output state. By following this stopping rule, it was determined that 13 clusters were required to describe the internal states of the network. The relationship between the different clusters and network output states is provided in Table 7A.

Relation of Hidden Unit Clusters to the Classical Rules. The key question to be addressed in this study is the relationship between hidden unit states (i.e., clusters) and the classical rules of the Study 1 decision tree. In Study 2, we discovered a unique mapping from clusters to rules, as every cluster mapped onto one and only one of the decision points in the Study 1 algorithm. Was this the case for the Study 3 algorithm?

As can be seen by examining Table 7B, there was not a unique mapping of clusters to classical rules for the Study 3 network. For example, Cluster 1 is composed of hidden unit activity vectors that are all produced by edible mushrooms. However, by examining the Cluster 1 row of Table 7B it can be seen that these edible mushrooms could not be identified by applying only one of the decision tree rules. Some of the mushrooms would be classified by Rule 1 Edible, others by Rule 2 Edible, and still others by Rule 3 Edible. Similar cases can be made for most of the other clusters for this network. Whatever rules are being used by the network, they are different from those described by the Study 1 decision tree, and therefore are also different from those embodied by the internal states of the Study 2 network.

Extracting a Decision Rule from the Network. While Table 7B indicates that the Study 3 network is not using the Study 1 algorithm, for completeness sake it is important to describe the procedure that it is using. We obtained such an account by carrying out the following three steps.

Table VII. (A) Crosstabulation table indicating the frequency of patterns classified in terms of (i) the cluster to which they belong and (ii) the network output that they produce. (B) Crosstabulation table indicating the frequency of patterns in terms of (i) the cluster to which they belong and (ii) problem type as determined by the Study 1 decision tree.

	Output State of the Network	
Cluster	Not Edible (Output = 0)	Edible 1
1	0	3288
2	0	224
3	976	0
4	0	408
5	36	0
6	264	0
7	1296	0
8	0	288
9	720	0
10	528	0
11	8	0
12	16	0
13	72	0

A

	Type of Response in terms of the Study 1 Decision Tree								
Cluster	Rule 1 Poisonous	Rule 1 Edible	Rule 2 Edible	Rule 2 Poisonous	Rule 3 Edible	Rule 5 Edible	Rule 4 Poisonous	Rule 5 Poisonous	Rule 4 Edible
1	0	776	2496	0	16	0	0	0	0
2	0	0	0	0	224	0	0	0	0
3	960	0	0	0	0	0	16	0	0
4	0	0	72	0	228	24	0	0	24
5	36	0	0	0	0	0	0	0	0
6	256	0	0	0	0	0	0	8	0
7	1296	0	0	0	0	0	0	0	0
8	0	24	264	0	0	0	0	0	0
9	720	0	0	0	0	0	0	0	0
10	528	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	8	0	0
12	0	0	0	0	0	0	16	0	0
1	0	0	0	72	0	0	0	0	0

B

Table VIII. Definite features for each of the 13 clusters from the Study 3 network

Feature	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6
N	3288	224	976	408	36	264
Cap Shape	~ Conical ~ Sunken	~ Bell ~ Conical ~ Sunken	~ Conical ~ Sunken	~ Conical ~ sunken	~ Bell ~ Conical ~ Sunken	~ Sunken
Cap Surface	~ Grooves	~ Fibrous ~ Grooves	~ Grooves	~ Grooves	Scaly~ Fibrous	
Cap Color	~ Buff ~ Cinnamon ~ Green ~ Pink ~ Purple	~ Green ~ Purple ~ White ~ Yellow	Brown or Gray or White or Yellow	Brown or Cinnamon or Gray or White	Brown or Cinnamon	Brown or White
Bruises		Bruises	No Bruises	No Bruises	No Bruises	Bruises
Odor	Almond or Anise or None	None	Creosote or Foul or None	None	Musty	None or Pungent
Gill Attachment	Free	Free	Free	Attached or Free	Attached or Free	Free
Gill Spacing	Close or Crowded	Close	Close or Crowded	Close or Crowded	Close	Close or Crowded
Gill Size		Broad			Broad	Narrow
Gill Color	~ Buff ~ Green ~ Orange ~ Red ~ Yellow	Red or White	~ Black ~ Buff ~ Green ~ Orange ~ Red ~ Yellow	~ Black ~ Buff ~ Chocolate ~ Green ~ Purple ~ Red	White or Yellow	Black or Brown or Pink or White
Stalk Shape		Enlarging	Enlarging	Enlarging	Enlarging	Enlarging
Stalk Surface Above Ring	~ Silky	Smooth	Silky or Smooth	~ Scaly	Silky	Smooth
Stalk Surface Below Ring	~ Silky	Smooth	~ Fibrous	~ Scaly	Scaly	Smooth
Stalk Color Above Ring	Brown or Grey or Pink or White	Red or White	Brown or Buff or Pink or White	Orange or White	Cinnamon	White
Stalk Color Below Ring	Brown or Grey or Pink or White	Red or White	Brown or Buff or Pink or White	Brown or Orange or White	Cinnamon	White
Veil Type	Partial	Partial	Partial	Partial	Partial	Partial
Veil Color	White	White	White	~ Yellow	White	White
Ring Number	One or Two	Two	One	One or Two	None	One

Table VIII. Continued

Feature	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6	
N	3288	224	976	408	36	264	
Population	∼ Clustered	Clusteredr or Several or Solitary	∼ Abundant ∼ Clustered ∼ Numerous	∼ Abundant ∼ Solitary	Clustered	∼ Abundant ∼ Numerous ∼ Solitary	
Habitat	Grasses or Meadows or Paths or Woods	Paths or Waste	Grasses or Paths or Woods	Grasses or Leaves	Woods	Grasses or Leaves or Urban	
Cap Shape	∼ Bell ∼ Conical ∼ Sunken	∼ Conical	∼ Bell ∼ Conical ∼ Sunken	Convex or Flat	∼ Convex ∼ Sunken	∼ Conical ∼ Sunken	Bell or Flat
Cap Surface	∼ Fibrous ∼ Grooves	∼ Grooves	∼ Fibrous ∼ Grooves	∼ Grooves	Scaly	∼ Grooves ∼ Smooth	∼ Fibrous ∼ Grooves
Cap Color	Brown or Buff or Grey or Red or White	∼ Buff ∼ Cinnamon ∼ Pink ∼ Red	∼ Cinnamon ∼ Green ∼ Pink ∼ Purple ∼ Yellow	Gray or Pink or White or Yellow	Yellow	Brown or Yellow	Buff or Pink or White
Bruises				No Bruises	No Bruises	No Bruises	Bruises
Odor	Foul or Spicy	Anise or None	Fishy or Foul	Creosote or Foul	None	None	None
Gill Attachment	Free	Attached or Free	Free	Free	Free	Free	Free
Gill Spacing	Close	Close or Crowded	Close	Close	Crowded	Close	Close
Gill Size					Narrow	Narrow	Broad
Gill Color	Buff or Chocolate or Pink or	∼ Buff ∼ Green ∼ Red	Buff or Chocolate or Pink or	Brown or Chocolate	White or Yellow	White	Gray or Green or White
Ring Type	Evanescent or Pendant	Evanescent or Pendant	Evanescent or Large or Pendant	Evanescent or Pendant	None	Pendant	
Spore Print Color	Black or Brown or Purple or White	White	Black or Brown or Chocolate or White	Orange or White or Yellow	White White	Black or Brown or	

Table VIII. Continued

Feature	Cluster 7	Cluster 8	Cluster 9	Cluster 10	Cluster 11	Cluster 12	Cluster 13
N	1296	288	720	528	8	16	72
	White		white	Or Gray or Pink or Purple			
Stalk Shape	Tapering		Tapering	Enlarging	Enlarging	Enlarging	Enlarging
Stalk Surface Above Ring	~ Scaly	Smooth	Silky or Smooth	~ Fibrous ~ Scaly	Scaly	Silky	Smooth
Stalk Surface Below Ring	~ Scaly	~ Scaly ~ Silky	~ Scaly	~ Fibrous ~ Scaly	Scaly	Scaly	Smooth
Stalk Color Above Ring	Pink or White	Orange or White	Pink or White	Brown or Buff or Pink or White	Yellow	White	White
Stalk Color Below Ring	Pink or White	Orange or White	Pink or White	Brown or Buff or Pink or White	Yellow	Yellow	White
Veil Type	Partial	Partial	Partial	Partial	Partial	Partial	Partial
Veil Color	White	~ Yellow	White	White	Yellow	White	White
Ring Number	One	One	One	One	One	One	Two
Ring Type	Evanescent or Pendant	Flaring or Pendant	Evanescent Or Pendant	Large or Pendant	Evanescent	Evanescent	Pendant
Spore Print Color	Chocolate or White	~ Green ~ Purple ~ White ~ Yellow	Chocolate or White	Black or Brown or Chocolate	White	White	Green
Population	Scattered or Several	Clustered or Several or Solitary	Scattered or Several	Scattered Or Several Or Solitary	Clustered	Several	Several
Habitat	~ Meadows ~ Waste	Leaves or Urban or Woods	~ Meadows ~ Waste	Grasses or Woods	Leaves	Woods	Grasses or Meadows

First, we identified the definite features associated with each cluster, using the same method that was used in Study 2. Recall that a definite feature is an input feature that is shared by all of the members of the same cluster. As was the case for Study 2, we were able to identify a rich set of definite features for each of the 13 clusters, as can be seen in Table 8.

The second step was to recode each column in Table 8 in a numerical format. When all of the possible values of the 21 different mushroom features are considered (i.e., all of the different values in the second column of Table 1), there are

119 different features that could be present or absent for a cluster. We took each column of Table 8 and generated a binary code for the features that it represented. If a feature could be present in a cluster, then we coded it with a value of 1. If a feature could not be present in a cluster, then we coded it with a value –1. As a result, we produced 13 different binary vectors, each of which was 119 features long.

The third step was to perform a discriminant analysis using these 13 different vectors. Each of the 119 features in the vector was used a predictor. The predicted variable was network response – whether the feature vector was associated with mushrooms that were edible or not. When this analysis was performed, the following discriminant function was produced:

$$\begin{aligned} Y \;=\; & 30^{*}(capcolor = cinnamon) + 55^{*}(odor = anise) \\ & -8^{*}(gillcolor = white) + 19^{*}(stalkcolorabovering = white) \\ & +11^{*}(ringtype = evanescent) - 8^{*}(habitat = meadows) \\ & -16^{*}(habitat = woods) - 11 \end{aligned}$$

This function is a linear weighting of 7 different features, each of which is either true (+1) or false (–1) of a cluster. This function, derived from the internal states of the network, will correctly classify all of the mushrooms. If the function returns a negative value, then the mushroom is edible. If the function returns a positive value, then the mushroom is not edible. While this function is a perfect classifier of mushroom type, it is not the same as the decision tree algorithm which was discovered in Study 1, and which was inserted into a PDP network in Study 2. In other words, the classical algorithm that was recovered from the Study 2 network is not the only procedure that could be used to solve the mushroom problem.

8. General Discussion

In Study 1, we used standard machine learning techniques to derive a classical decision tree for the mushroom problem. In Study 2, we used extra output learning to attempt to insert this decision tree into a network of value units. Our analysis of the trained network indicated a unique mapping from internal network states (i.e., clusters of hidden unit activities) to a set of productions that were equivalent to the Study 1 algorithm. This mapping is an instance of an intertheoretic reduction between the Study 1 algorithm and the Study 2 network. In Study 3, we determined whether other network algorithms – procedures that did not map onto the Study 1 decision tree – were possible. We found that a network trained on the problem without using extra output learning classified the mushrooms using a procedure that was unlike the method that was discovered in Study 1. This indicated that the extra output learning technique used in Study 2 was responsible for inserting the classical algorithm into the network of value units.

What are the implications of this finding for the architectural debate that is being conducted in cognitive science? The main implication is that one cannot assume that classical models and connectionist networks are fundamentally different, because we have demonstrated that one can take one and translate it into the other. In other words, the main result of the current paper is to demonstrate that at the algorithmic level it is possible to have a classical model that is exactly equivalent to a PDP network.

This is not to say, of course, that every classical model is algorithmically equivalent to a PDP network, or vice versa. However, given that we have shown that in some cases the two types of theories can be equated at this level, if one wants to say that a PDP model is different from a classical theory, then one must justify this claim by interpreting the network. The interpretation of the Study 2 network has indicated that something that doesn't look very much like a classical algorithm can in fact be precisely equivalent to that algorithm.

To complete a discussion from the introduction, the final level of analysis to consider for a comparison between classical and connectionist models is the level of implementation. At this level, the question that is addressed is "What physical properties are required to build the functional architecture into a physical device?" (Dawson, 1998). This level has been the source of a great deal of controversy in the debate between these two approaches to cognitive science. On the one hand, many proponents of connectionism have argued that PDP models are more biologically plausible than are classical systems (Clark, 1989; Clark, 1993; Dreyfus and Dreyfus, 1988; McClelland, Rumelhart, and Hinton, 1986). On the other hand, classical supporters have claimed that if connectionist models are to be taken as biological accounts, then they are not part of cognitive science because they do not appeal to a cognitive vocabulary (Broadbent, 1985; Fodor and Pylyshyn, 1988; Pylyshyn, 1991). "The problem with Connectionist models is that all the reasons for thinking that they might be true are reasons for thinking that they couldn't be psychology" (Fodor and Pylyshyn, 1988, p. 66).

However, there are many reasons to delay a comparison between the two approaches at the implementational level. First, many researchers have pointed out that many properties of PDP networks are not biologically plausible (Crick and Asanuma, 1986; Douglas and Martin, 1991; Smolensky, 1988). Second, many analyses of connectionism indicate (at the very least) that it is unclear whether PDP networks are to be understood as implementational theories or as cognitive theories (Broadbent, 1985; Dawson, 1998; Rumelhart and McClelland, 1985). Third, it has been shown that novel cognitive (as opposed to implementational) theories can be extracted from connectionist networks (Dawson et al., 1997).

While our position is that an implementational comparison of classical and connectionist models is premature, the results that we have reported above present an interesting opportunity for future research. Our simulations have demonstrated that extra output learning can be used to translate a particular classical theory into one connectionist network. An area that remains to be explored is determining the extent to which extra output learning can be used as a general technique for theory translation. Can all classical theories be translated into PDP networks? Or are there limitations on the kinds of translation that are possible? If at some future point it is established that PDP models are more appropriate for cognitive science because of implementational or architectural considerations, then classical cognitive science may be in need of answers to such questions about theory translation.

Acknowledgements

This research was supported by an NSERC Research Grant to MRWD, a Killam Fellowship awarded to DAM, and NSERC scholarships awarded to DBM and LW.

Address correspondence to: Dr. Michael Dawson, Biological Computation Project, Department of Psychology, University of Alberta, Edmonton, Alberta, CANADA T6G 2E9. E-mail: mike@psych.ualberta.ca.

References

Abu-Mostafa, Y. S. (1990) 'Learning from hints in neural networks', *Journal of Complexity*, 6, pp. 192–198.

Aldenderfer, M. S. and Blashfield, R. K. (1984), *Cluster Analysis*. (Vol. 07-044), Beverly Hills, CA: Sage Publications.

Andrews, R., Diederich, J. and Tickle, A. B. (1995), 'A survey and critique of techniques for extracting rules from trained artificial neural networks', *Knowledge-Based Systems*, 8, pp. 373–389.

Bechtel, W., and Abrahamsen, A. (1991), *Connectionism and the Mind*, Cambridge, MA: Basil Blackwell.

Berkeley, I. S. N., Dawson, M. R. W., Medler, D. A., Schopflocher, D. P. and Hornsby, L. (1995), Density plots of hidden value unit activations reveal interpretable bands, *Connection Science*, pp. 167–186.

Born, R. (1987), *Artificial intelligence: The case against*, London: Croom Helm.

Broadbent, D. (1985), 'A question of levels: Comment on McClelland and Rumelhart', *Journal of Experimental Psychology: General*. 114, pp. 189–192.

Caruana, R. and de Sa, V. R. (1997), 'Promoting poor features to supervisors: Some inputs work better as outputs', in M. C. Mozer, M. I. Jordan and T. Petsche (eds.), *Advances in Neural Information Processina Systems 9*, Cambridge, MA: MIT Press.

Churchland, P. M. (1985), 'Reduction. qualia, and the direct introspection of brain states', *The Journal of Philosophy. LXXXII*, pp. 8–28.

Churchland, P. M. (1988), *Matter and consciousness. Revised edition*, Cambridge, MA: MIT Press.

Churchland, P. M. (1995), *The engine of reason. the seat of the soul*, Cambridge, MA: MIT Press.

Churchland, P. S., Koch, C. and Sejnowski, T. J. (1990), 'What is computational neuroscience?', in E. L. Schwartz (ed.), *Computational Neuroscience*, Cambridge, MA: MIT Press.

Churchland, P. S. and Sejnowski, T. J. (1989), 'Neural representation and neural computation', in L. Nadel, L. A. Cooper, P. Culicover, and R. M. Harnish (eds.), *Neural Connections. Mental Computation*, Cambridge, MA: MIT Press, pp 15–48.

Churchland, P. S. and Sejnowski, T. J. (1992), *The computational brain*, Cambridge, MA: MIT Press.

Clark, A. (1989), *Microcoanition*, Cambridge, MA: MIT Press.

Clark, A. (1993), *Associative engines*, Cambridge, MA: MIT Press.

Crick, F. and Asanuma, C. (1986), 'Certain aspects of the anatomy and physiology of the cerebral cortex', in J. McClelland and D. E. Rumelhart (eds.), *Parallel Distributed Processing* (Vol. 2), Cambridge, MA: MIT Press.

Dawson, M. R. W. (1990), ‘Training networks of value units: Learning in PDP systems with nonmonotonicactivation functions’, *Canadian Psychology* 31(4), pp. 391.

Dawson, M. R. W. (1991), ‘The how and why of what went where in apparent motion: Modeling solutions to the motion correspondence process’, *Psychological Review* 98, pp 569–603.

Dawson, M. R. W. (1998), *Understanding Cognitive Science*. Oxford, UK: Blackwell.

Dawson, M. R. W., Medler, D. A. and Berkeley, I. S. N. (1997), ‘PDP networks can provide models that are not mere implementations of classical theories. Philosophical Psychology. 10, 25-40. Dawson, M. R. W. and Schopflocher, D. P. (1992a), ‘Autonomous processing in PDP networks’, *Philosophical Psychology*. 5, pp. 199–219.

Dawson, M. R. W. and Schopflocher, D. P. (1992b), ‘Modifying the generalized delta rule to train networks of nonmonotonic processors for pattern classification’, *Connection Science* 4, pp. 19–31.

Dawson, M. R. W. and Shamanski, K. S. (1994), ‘Connectionism, confusion and cognitive science’, *Journal of Intelligent Systems*. 4, pp. 215–262.

Dawson, M. R. W., Shamanski, K. S. and Medler, D. A. (1993), *From connectionism to cognitive science*. Paper presented at the Fifth University of New Brunswick Symposium on Artificial Intelligence, Fredericton, NB.

Douglas, R. J. and Martin, K. A. C. (1991), ‘Opening the grey box’, *Trends In Neuroscience* 14, pp. 286–293.

Dreyfus, H. L. and Dreyfus, S. E. (1988), ‘Making a mind versus modeling the brain. Artificial intelligence back at the branchpoint’, in S. Graubard (ed.), *The Artificial Intelligence Debate*, Cambridge, MA: MIT Press.

Elman, J. (1990), ‘Finding structure in time’, *Cognitive science* 14, pp. 179–211.

Everitt, B. (1980), *Cluster Analysis* New York: Halsted.

Fodor, J. A. and McLaughlin, B. P. (1990), ‘Connectionism and the problem of systematicity: Why Smolensky’s solution doesn’t work’, *Cognition* 35, pp. 183–204.

Fodor, J. A. and Pylyshyn, Z. W. (1988), ‘Connectionism and cognitive architecture’, *Cognition* 28, pp. 3–71.

Gallant, S. I. (1993), *Neural network learning and expert systems*, Cambridge, MA: MIT Press.

Gailmo, 0. and Carlstrom, J. (1995), ‘Some experiments using extra output learning to hing multilayer perceptrons’, in L. F. Niklasson and M. B. Boden (eds.), *Current Trends in Connectionism – Proceedings of the 1995 Swedish Conference on Connectionism*, Hillsdale, NJ: Lawrence Erlbaum Associates, pp. 179–190.

Garson, J. W. (1994), ‘No representations without rules: The prospects for a compromise between paradigms in cognitive science’, *Mind and Language* 9, pp. 25–37.

Graubard, S. (1988), *The artificial intelligence debate*, Cambridge, MA: MIT Press.

Hadley, R. F. (1994a), ‘Systematicity in connectionist language learning’, *Minds and Machines* 3, pp. 183–200.

Hadley, R. F. (1994b), ‘Systematicity revisited: Reply to Christiansen and Chater and Niclasson and van Gelder’, *Mind and Language* 9, pp. 431–444.

Hadley, R. F. (1997), ‘Cognition, systematicity, and nomic necessity’, *Mind and Language* 12, pp. 137–153.

Hadley, R. F. and Hayward, M. B. (1997), ‘Strong semantic systematicity from Hebbian connectionist learning’, *Minds and Machines* 7, pp. 1–37.

Hanson, S. J. and Burr, D. J. (1990), ‘What connectionist models learn: Learning and representation in connectionist networks’, *Behavioral and Brain Sciences* 13, pp. 471–518.

Haugeland, J. (1985), *Artificial intelligence: The very idea*, Cambridge, MA: MIT Press.

Hecht-Nielsen, R. (1987), *Neurocomputing*, Reading, MA: Addison-Wesley.

Hinton, G. E. (1986), *Learning distributed representations of concepts*. Paper presented at the the 8th Annual Meeting of the Cognitive Science Society, Ann Arbor, MI.

Hooker, C. A. (1979), ‘Critical notice: R.M. YoshidaŠs Reduction in the Physical Sciences’, *Dialogue* 18, pp. 81–99.

Hooker, C. A. (1981), ‘Towards a general theory of reduction’, *Dialogue* 20, pp. 38–59, 201–236, 496–529.

Hopcroft, J. E. and Ullman, J. D. (1979), *Introduction to Automata Theorv. Languages. and Computation*, Reading. MA: Addison-Wesley.

Horgan, T. and Tienson, J. (1996), *Connectionism and the philosophy of psychology*, Cambridge, MA: MIT Press.

Kilian, J. and Siegelmann, H. T. (1993), *On the power of sigmoid neural networks*. Paper presented at the Proceedings of the Sixth ACM Workshop on Computational Learning Theory.

Kremer, S. C. (1995), ‘On the computational powers of Elman-style recurrent networks’, *IEEE Transactions on neural networks* 6, pp. 1000–1004.

Lachter, J. and Bever, T. G. (1988), ‘The relation between linguistic structure and associative theories of language learning - A constructive critique of some connectionist learning models’, *Cognition* 28, pp. 195–247.

Lincoff, G. H. (1981), *National Auduboii Society field guide to North American mushrooms*, New York: Alfred A. Knopf Publishers.

Marr, D. (1982), *Vision*, San Francisco, CA. W.H. Freeman.

McCaughan, D. B. (1997, June 9–12), *On the properties of periodic perceptrons*. Paper presented at the IEEE/INNS International Conference on Neural Networks (ICNN’97), Houston, TX.

McClelland, J. (1992), ‘Can connectionist models discover the structure of natural language?’, in R. Morelli, W. M. Brown, D. Anselmi, K. Haberlandt, and D. Lloyd (eds.), *Minds, Brains. and Computers: Perspectives in Cognitive Science and Artificial Intelligence*, Norwood, NJ: Ablex.

McClelland, J. L., Rumelhart, D. F. and Hinton, G. E. (1986), ‘The appeal of parallel distributed processing’, in D. Rumelhart and J. McClelland (eds.), *Parallel Distributed Processing* (Vol. 1), Cambridge, MA: MIT Press.

McCloskey, M. (1991), ‘Networks and theories: The place of connectionism in cognitive science’, *Psychological Science* 2, pp. 387–395.

McCulloch, W. S. and Pitts, W. (1943), ‘A logical calculus of the ideas immanent in nervous activity’, *Bulletin of Mathematical Biophysics* 5, pp. 115–133.

Medler, D. A. (1998), *The crossroads of connectionism: Where do we go from here?* Unpublished Doctoral dissertation, University of Alberta, Edmonton, AB.

Michie, D., Speigelhalter, D. J. and Taylor, C. C. (1994), *Machine learning, neural and statistical classification*. New York, NY: Ellis Horwood.

Milligan, G. W. and Cooper, M. C. (1985), ‘An examination of procedures for determining the number of clusters in a data set’, *Psychometrika* 50, pp. 159–179.

Minsky, M. (1972), *Computation: finite and infinite machines*. London: Prentice-Hall International.

Mozer, M. C. and Smolensky, P. (1989), ‘Using relevance to reduce network size automatically’, *Connection Science* 1, pp. 3–16.

Omlin, C. W. and Giles, C. L. (1996), ‘Extraction of rules from discrete-time recurrent neural networks’, *Neural networks* 9, pp. 41–52.

Pinker, S. and Prince, A. (1988), ‘On language and connectionism: Analysis of a parallel distributed processing model of language acquisition’, *Cognition* 28, pp. 73–193.

Pylyshyn, Z. W. (1984), *Computation and cognition*, Cambridge, MA.: MIT Press.

Pylyshyn, Z. W. (1991), ‘The role of cognitive architectures in theories of cognition’, in K. VanLehn (ed.), *Architectures For Intelligence*, Hillsdale, NJ: Lawrence Eribaum Associates, pp 189–223.

Quinlan, J. R. (1986), ‘Induction of decision trees’, *Machine Learning* 1, pp. 81–106.

Ramsey, W., Stich, S. P. and Rumelhart, D. E. (1991), *Philosophy and connectionist theory*, Hillsdale, NJ: Lawrence Erlbaum Associates.

Ripley, B. D. (1996), *Pattern recognition and neural networks*. Cambridge, UK: Cambridge University Press.

Rumeihart, D. E., Hinton, G. E. and Williams, R. J. (1986), 'Learning representations by back-propagating errors', *Nature* 323, pp. 533–536.

Rumelhart, D. E. and McClelland, J. L. (1985), 'Levels indeed! A response to Broadbent', *Journal of Experimental Psychology: General* 114, pp. 193–197.

Schlimmer, J. S. (1987), *Concept acquisition through representational adjustment*. Unpublished Doctoral dissertation, University of California Irvine, Irvine, CA.

Schneider, W. (1987), 'Connectionism: Is it a paradigm shift for psychology?', *Behavior Research Methods, Instruments and Computers* 19, pp. 73–83.

Seidenberg, M. (1993), 'Connectionist models and cognitive theory', *Psychological Science* 4, pp. 228–235.

Siegelman, H. T. and Sontag, E. D. (1991), 'Turing computability with neural nets', *Applied Mathematics Letters* 4, pp. 77–80.

Siegelmann, H. T. and Sontag, E. D. (1995), 'On the computational power of neural nets', *Journal of Computer and System Sciences* 50, pp. 132–150.

Sigelmann, H. T. (1999), *Neural Networks and Analog Computation: Beyond the Turing Limit*, Boston, MA: Birkhauser.

Smith, B. C. (1996), *On the Origin of Objects*, Cambridge, MA: MIT Press.

Smolensky, P. (1988), 'On the proper treatment of connectionism', *Behavioural and Brain Sciences* 11, pp. 1–74.

Stork, D. G. (1997), 'Scientist on the set: An interview with Marvin Minsky', in D. G. Stork (ed.), *HAL's Legacy: 2001's Computer as Dream and Reality*, Cambridge, MA: MIT Press, pp. 15–32.

Suddarth, S. C. and Kergosien, Y. L. (1990), 'Rule-injection hints as a means of improving network performance and learning time', in L. B. Almeida and C. J. Wellekens (eds.), *Neural Networks. Lecture Notes in Computer Science* (Vol. 412), Berlin: Springer Verlag, pp. 120–129.

Suddarth, S. C., Sutton, S. A. and Holden, A. D. C. (1988), *A symbolic-neural method for solving control problems*, Paper presented at the IEEE International Conference on Neural Networks, San Diego, CA.

VanLehn, K. (1991), *Architectures for intelligence*, Hillsdale, NJ: Lawrence Erlbaum Associates.

Von Eckardt, B. (1993), *What is cognitive science?*, Cambridge, MA: MIT Press.

Williams, R. and Zipser, D. (1989), 'A learning algorithm for continually running fully recurrent neural networks', *Neural Computation* 1, pp. 270–280.

Yu, Y.-H. and Simmons, R. F. (1990), 'Extra output based learning', *Proceedings of the International Joint Conference on Neural Networks* (IJCNN-90) 3, pp. 161–166.