ELSEVIER

Journal of Cognitive Systems Research 2 (2001) 207–231

Cognitive Systems RESEARCH

www.elsevier.com/locate/cogsys

A parallel distributed processing model of Wason's selection task

Action editor: Steve J. Hanson

Jacqueline P. Leighton[a,*], Michael R.W. Dawson[b]

[a]*Centre for Research in Applied Measurement and Evaluation, Faculty of Education, 6-110 Education North Building, University of Alberta, Edmonton, Alberta, Canada T6G 2G5*

[b]*Biological Computation Project, Department of Psychology, University of Alberta, Edmonton, Alberta, Canada T6G 2E9*

Received 1 September 2000; received in revised form 10 March 2001; accepted 17 June 2001

Abstract

Architectural accounts of cognitive performance are important to explore because they provide the infrastructure for algorithmic theories of cognition [Dawson, M.R.W. (1998). *Understanding cognitive science*. Malden, MA: Blackwell]. Three parallel distributed processing (PDP) networks were trained to generate the 'p', the 'p and not-q' and the 'p and q' responses, respectively, to the conditional rule used in Wason's selection task [Wason, P.C. (1966). Reasoning. In: Foss, B.M. (Ed.), *New Horizons in Psychology*, London, Penguin]. Afterward, each trained network was analyzed for the algorithm it developed to learn the desired response to the task. Analyses of each network's solution to the task suggested a 'specialized' algorithm that focused on card location. For example, if the desired response to the task was found at card 1, then a specific set of hidden units detected the response. In addition, we did not find support that selecting the 'p' and 'q' response is less difficult than selecting the 'p' and 'not-q' response. Human studies of the selection task usually find that participants fail to generate the latter response, whereas most easily generate the former. We discuss how our findings can be used to (a) extend our understanding of selection task performance, (b) understand existing algorithmic theories of selection task performance, and (c) generate new avenues of study of the selection task. © 2001 Elsevier Science B.V. All rights reserved.

1. Introduction

No other task has spurred as much research into human reasoning as has Wason's (1966) selection task (Evans, Newstead, & Byrne, 1993). Fig. 1 illustrates the task, which involves presenting a participant with a conditional rule in the form of *If P then Q* and four cards displaying instances of a *p*, a *not-p*, a *q*, and a *not-q*. In the actual task, each card contains a letter on one side and a number on the other side. Participants are instructed to test the truth or falsity of the rule by selecting the fewest cards possible from the set of four. Although participants can see only side of each card, they are told that each card's flip side contains information that might be useful in testing the rule.

According to propositional logic, only the pairing of the 'p' (i.e., the antecedent) with the 'not-q' (i.e., the negation of the consequent) falsifies and, there-

*Corresponding author. Tel.: +1-780-492-1167; fax: +1-780-492-0001.

E-mail address: jacqueline.leighton@ualberta.ca (J.P. Leighton).

1389-0417/01/$ – see front matter © 2001 Elsevier Science B.V. All rights reserved.
PII: S1389-0417(01)00035-3

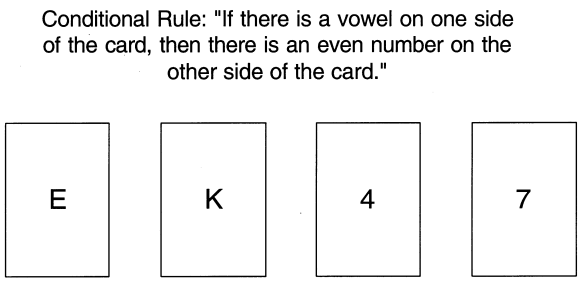

Fig. 1. Wason's card selection task.

fore, conclusively tests the rule (Garnham & Oakhill, 1994). Typically only 10% of participants select cards corresponding to 'p' and 'not-q', however. Most participants select either the 'p' alone or the 'p' and the 'q' (for a complete review of the selection task, the reader is referred to Evans et al., 1993).

Why do so many participants fail to respond logically to the selection task? This is a question that has spurred numerous studies and a host of theories (e.g., Cheng & Holyoak, 1985, 1989; Cosmides, 1989; Johnson-Laird & Byrne, 1991; Rips, 1994). Unfortunately, many of these theories are algorithmic and do not address the question of the kind of architecture that underlies performance. Specifying the architecture of cognitive performance is useful because it anchors algorithmic theories to more concrete descriptions of performance; 'black box' descriptions are precluded (Dawson, 1998).

Cheng and Holyoak (1985, 1989) account for participants' poor performance with a *pragmatic reasoning theory*. According to the theory, participants possess domain-specific schemata that, invoked in meaningful contexts, facilitate reasoning responses. Cheng and Holyoak (1985, 1989) suggest that participants perform poorly on the selection task because it lacks a meaningful context. In support, they have empirically shown that logical responses to the selection task increase when the task is framed in a meaningful context, such as when participants must decide which person, among four, stands in violation of a permission rule (see Liberman and Klar, 1996, for a discussion of how the permission context may change the nature of the task). Cosmides' (1989) *social contract theory* proposes a similar account of participants' poor performance, although the domain-specific schemata in this theory are invoked specifically in response to situations involving costs and benefits. Alternatively, Johnson-Laird's *mental models theory* (1983; Johnson-Laird & Byrne, 1991) proposes that any context that impairs participants' ability to consider counter-examples to the selection task's rule will hinder logical performance. Finally, Rips' (1994) *syntactic theory* proposes that participants fail to respond correctly to the selection task because the task calls for a sophisticated application of mental rules that many participants either do not have or have not yet mastered. One similarity among the first three theories — pragmatic reasoning theory, social contract theory, and mental models — is that each provides an algorithmic account of performance; that is, each theory specifies a procedural description of how participants solve the selection task. Rips' (1994) syntactic theory also provides an algorithmic account but, in addition, the theory specifies an *architectural* account of participants' performance — a system of productions that implements logical routines.

Rips' (1994) theory suggests that a system of productions underlies reasoning, but some critics have argued that his theory is unconvincing because it fails to reflect the *inductive* quality of human reasoning (e.g., Oaksford & Chater, 1993). For example, unlike pragmatic reasoning theory, social contract theory, and mental models theory, syntactic theory in general ignores the role of context in reasoning, and the non-monotonicity of reasoning (Byrne, 1989; Evans et al., 1993). According to some theorists, other architectures might offer more credible accounts of reasoning (e.g., Oaksford & Chater, 1993). In particular, a connectionist architecture might be more representative of human reasoning, as Shastri (1991) explains:

> *Connectionism offers an extremely efficient metaphor for reasoning where inference is reduced to spreading activation in a parallel network . . . . The connectionist approach suggests alternate formulations of information processing. Thus instead of viewing a knowledge-based system as a theorem prover or a production system, one may view it as a system that performs constraint satisfaction, energy minimization, or evidential and probabilistic reasoning* (*p.* 263).

Specifying the architecture of an algorithmic account of cognitive performance is important because it provides ''an account of the mental program-

ming language in which cognitive algorithms are written'' (Dawson, 1998, p. 159). Such an account serves to clarify theories and to extend our understanding of the cognitive performance under study. For instance, we can pursue questions such as ''how might domain-specific schemata be instantiated in the brain?'' or ''are rules or some other operation underlying performance?'' To begin answering such questions, we need to explore architectural accounts of reasoning alongside algorithmic accounts. Exploring functional architectural accounts of cognitive performance can bring us closer to more complete theories of cognition.

In this paper we explore whether a specific connectionist or parallel distributed processing (PDP) architecture, termed the *value unit* architecture, can extend our understanding of participants' performance on the selection task. Hence, we attempt to resolve the critique we levy against theories of selection task performance; namely, that they are not linked to a functional architecture. This research is exploratory since to our knowledge no one has simulated performance on the selection task using a connectionist architecture. Other investigators have used connectionist architectures to model other kinds of (inferential) problems but not specifically the selection task (e.g., Derthick, 1991; Shastri, 1991).

As a general overview of the paper, we first discuss why PDP networks can be used to explore an architectural account of selection task performance. Second, we provide a general introduction to PDP networks and in particular the value unit architecture, which is the specific architecture we use in the present studies. Third, we illustrate how three different value unit networks were trained to generate different responses to the selection task and how each network can help us understand participants' responses to this task. Finally, we discuss the results from all three networks and their relation to existing algorithmic accounts of performance on the selection task.

2. Why use PDP networks to explore an architectural account of performance on the selection task?

2.1. Pattern classification

There are two reasons why we would want to use PDP networks to extend our understanding of selection task performance. First, PDP networks learn to solve tasks by means of pattern classification or mapping input patterns to output responses. This method of learning to solve tasks is compatible with many of the algorithmic theories of selection task performance, such as pragmatic reasoning theory (Cheng & Holyoak, 1985). In fact, pattern classification accounts of reasoning in general have been proposed (e.g., Bechtel & Abrahamsen, 1991; Gobet & Simon, 1998; Goldstone & Barsalou, 1998). Pattern classification accounts of reasoning assume that people make sense of their environment by categorizing objects and events not only to make predictions about their (unseen) characteristics, but also to decide upon actions in light of the categorization. For example, Bechtel and Abrahamsen (1991), reiterating an idea proposed by Margolis (1987), suggest the following view of how reasoning might proceed according to the pattern classification view:

> *The recognition of one pattern constitutes an internal cue which, together with the external cues available from outside the system, facilitates yet another recognition. Thus, we work our way through a complex problem by recognizing something, and with the help of that result, recognizing something further.* (*p.* 141)

Viewing reasoning from the perspective of pattern classification complements theories such as Cheng and Holyoak's pragmatic reasoning theory (1985) and Cosmides' social contract theory (1989), which emphasize the inductive quality of reasoning.

Another important reason for employing PDP networks is that they characterize, albeit roughly, the kind of processing that occurs in the brain (Dawson, 1998). PDP networks are considered 'brain-like systems' in that they are built from inter-connected, simple processing units that can be used to classify patterns. We turn now to a description of PDP networks.

2.2. A PDP network of value units

A PDP network is a system of inter-connected, simple processing units that can be used to classify patterns presented to it. A PDP network is usually made up of three kinds of processing units: (a) *Input*

units encode the stimulus or activity pattern that the network will eventually classify; (b) *hidden units* detect features or regularities in the input patterns that can be used to determine classification decisions; and (c) *output units* represent the network's response to the input pattern; that is, the category to which the pattern is to be assigned on the basis of the features or regularities that have been detected by the hidden units. Processing units communicate by means of weighted connections. Fig. 2 provides an illustration of a typical network.

In most cases, a processing unit carries out three central functions: First, a processing unit computes the net input or the total signal that it receives from other units. A *net input function* is used to carry out this calculation. After the processing unit determines its net input, it transforms it into an internal level of activity, which typically ranges between 0 and 1. The internal activity level is calculated by means of an *activation function*. Finally, the processing unit determines the signal that needs to be sent to other units. This signal is created by applying an *output function* to the unit's internal activity. The most common output function is the identity function, suggesting that the signal sent out from a unit equals the unit's internal activity. (The reader is referred to Dawson (1998) for a more complete explication of the different functions.) A weighted connection acts as a communication channel between two processing units, and either amplifies or attenuates the signal being sent from one processing unit to another.

A network is not given a 'step by step' procedure for solving a desired task, but, is instead *trained* to solve the task. Consider a popular supervised learning procedure called the *generalized delta rule* (Rumelhart, Hinton & Williams, 1986). To train a network with this rule, one starts with a network (of a pre-specified number of hidden units) that has small, randomly assigned connection weights. The network is then 'developed' by presenting it a set of training patterns, each of which is associated with a desired response. To train a network to classify a pattern correctly, a pattern is presented to the network's input units, and the network generates a response to this stimulus using its existing connection weights. The network's response is then compared against the desired output (i.e., the correct response) and an error value is computed for each of the network's output units. This error value is then fed backwards through the network, and it is used to modify connection weights in such a way that the next time the pattern is presented to the network, the network's output errors will be smaller. By repeating this procedure a large number of times for each pattern in the training set, the network's response errors for each pattern can be reduced to near zero. At the end of this procedure, the network will have a very specific pattern of connectivity (in comparison

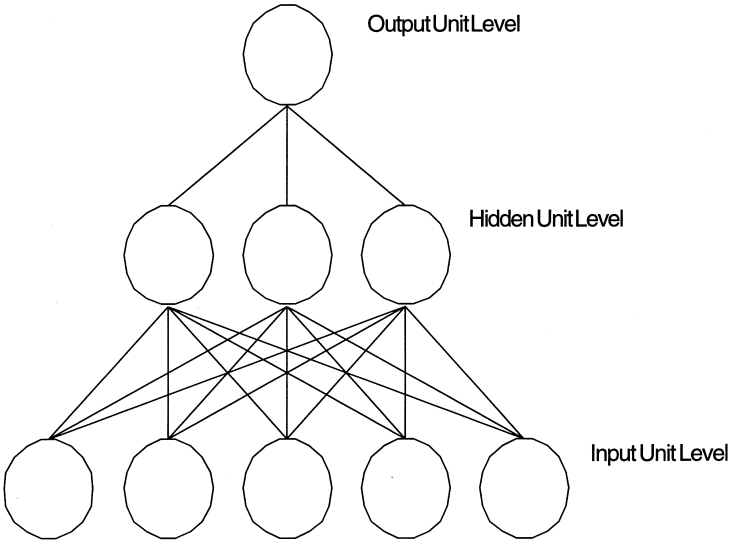

Fig. 2. Illustration of a typical PDP network, including layer of input units, hidden units, and output units.

to its random start) and will have *learned* to perform the desired stimulus/response pairing (if it is possible for such a pairing to be learned).

A number of different versions of the generalized delta rule exist, each designed to train networks whose processors have specific properties. For instance, one form of the generalized delta rule is applied when a logistic equation is used as an activation function (Rumelhart et al., 1986). A modified version of the generalized delta rule can be used to train networks of *value units* (Dawson & Schopflocher, 1992). A value unit is a processor that uses a Gaussian equation for its activation function.

2.3. Problem difficulty and a PDP network's algorithm for problem solving

It is possible to evaluate a problem's difficulty by the extent of a network's requirements for learning to solve the problem. For instance, the number of hidden units that a network requires to solve a problem is indicative of the problem's difficulty (Dawson, 1998). Hidden units allow connectionist networks to solve *linearly nonseparable* problems. Linearly nonseparable problems are difficult to solve, compared to linearly separable problems, because they require the network to divide the pattern space into multiple *decision regions*. In contrast, linearly separable problems require the network to make a single division in the pattern space so as to create two decision regions. Linearly nonseparable problems can be solved by networks that have a layer of hidden units. Each hidden unit can be viewed as creating a cut in the pattern space. Hence, the greater the number of hidden units required by a network to solve a problem, the greater the number of decision regions or 'cuts' in the pattern space demanded by the task.

Later in the paper, we will show how the activity within hidden units can be analyzed and interpreted to uncover the algorithm by which a network learns to solve a particular task. Interpreting hidden unit activity informs us of the specific features that define the decision regions created by the network to solve the task. Although analyzing hidden unit activity does not necessarily inform the question of problem or task difficulty, hidden unit activity does inform the question of *how* the network solved the task; that is, what are the important pattern features that define the decision regions and lead to the network's correct responses?

At first glance it might appear uninteresting to investigate whether a PDP network learns the desired input/output response to the selection task since we know that PDP networks, as universal Turing machines, should have little difficulty learning the task (Dawson, 1998). Our purpose, however, is not simply to see whether we can train a network to solve the selection task but, more importantly, to explore the algorithm or *how* a trained network solves the task. The exploration is not trivial since we cannot predict the algorithm a connectionist network will develop to solve a task. Exploring how a trained network solves the selection task can extend our understanding of the task and its existing theories. It is because of our interest in exploring how a trained network solves the task that we use the value unit architecture, whose hidden units often exhibit properties which render them interpretable (e.g., Berkeley, Dawson, Medler, Schopflocher & Hornsby, 1995; Dawson, 1998; Dawson, Medler & Berkeley, 1997). Next, we train a value unit network to generate a response that human participants commonly make to the selection task, but is incomplete.

3. Network 1: selection of the 'p' card

The goal of the first study was to train a value unit PDP network to generate the 'p' card in response to the selection task. We trained this first network to generate the 'p' card for two reasons: First, participants typically select the 'p' card alone in response to the selection task (see Evans et al., 1993). Second, we wanted to train a network to generate a comparatively simple response, the selection of only one card, before training a network to generate a more difficult response. The purpose of training this network (and others which will be described later in studies 2 and 3) is to determine the algorithm by which the network learns to generate a simple, but incomplete response, to the task. Ultimately, we want to examine the trained network's algorithm to see what it can teach us about performance on the selection task and existing algorithmic theories of the task. In the next section we discuss how we trained the network.

3.1. Method

3.1.1. PDP version of Wason's selection task: network architecture

When human participants are presented with the selection task, they already have a great deal of knowledge about its components. For instance, participants come to the task with knowledge of (a) the connective 'if then', (b) different kinds of numbers (i.e., odd versus even), and (c) different letters (i.e., vowels versus consonants). In contrast, PDP networks do not start with this kind of knowledge. PDP networks must first be given this preliminary information via some format that the network can process. Furthermore, the responses or behavior the network generates at the end of training should be broad or general enough (i.e., applicable across a large set of distinct patterns) to be of interest to psychologists. Hence, we needed to first find a way to encode the task for the network and, second, to create a sufficient number of input stimuli so that the network could be trained on a large number of patterns.

To solve these issues, a binary code was developed that allowed a representation of both the task's conditional rule and the four cards using 16 input units. Four inputs units were used to represent the rule. The first two input units reflected the antecedent of the rule, whereas the last two units reflected the consequent of the rule (see Table 1). Four sets of three input units were then used to represent the card categories. The first two input units of each set were used to represent the card's category membership, whereas the last unit of each set was used to represent its specific instance. Using this encoding scheme, a training set was developed that included eight different conditional rules and eight different instances of card categories (two vowels, two consonants, two even numbers, and two odd numbers). These card instances were crossed with 24 unique orders generated from assembling four 'cards' in all possible combinations. This latter step led to 384 unique orders of card values, which were then crossed with each of the eight rules to produce a final training set of 3072 input patterns.

We believe that our encoding scheme captures the underlying structure of the selection task. First, our binary coding scheme distinguishes exemplars from their categories in a form that is easily available to the network. As illustrated in Table 1, each card is represented with three input units in such a way that the first two input units represent the card's category, and the third input represents the card's exemplar of that category. For instance, the first two zeros in the code '000' indicate that this string is a 'vowel' while the third zero indicates that, specifically, this string is a letter 'A.'[1]

Table 1
Binary coding of conditional rule types and cards used to train networks and generate input patterns

Text form	Binary form
Rules:	
If vowel then even number	0 0 1 0
If vowel then odd number	0 0 1 1
If consonant then even number	0 1 1 0
If consonant then odd number	0 1 1 1
If even number then vowel	1 0 0 0
If even number then consonant	1 0 0 1
If odd number then vowel	1 1 0 0
If odd number then consonant	1 1 0 1
Cards:	
A	0 0 0
E	0 0 1
J	0 1 0
K	0 1 1
4	1 0 0
6	1 0 1
5	1 1 0
7	1 1 1

[1] Although the encoding of 'vowel' (i.e., 00) is on the surface more similar to the encoding of 'even number' (i.e., 10) than to 'odd number' (i.e., 11), this surface similarity should not bias the network's solution. The reason for this is that our training set included all possible permutations of rules and associated solutions; hence, that the rules and 'cards' share some surface similarity among a portion of the patterns is offset by the lack of similarity shared among the remaining portion of patterns. For example, suppose the network's task is to learn to select the 'p' (antecedent) and the 'q' (consequent) in response to the rules it is presented (see study 3 in this paper). In response to one form of the rule, ''if vowel then even'' (0010), the network would learn to select the vowel card (00) and the even card (10). In response to another rule, ''if vowel then odd'' (0011), the network would learn to select the vowel card (00) and the odd card (11). Hence, any surface similarity between some rules and some cards should not bias the network's solution of the task because the network learns to solve the task by responding to the full set of rules and cards in the training set.

Second, because the network was examined after it was presented a large number of training patterns, it had the opportunity to determine for itself the underlying nature of the task. For example, all of the networks that we trained learned to ignore the last 'bit' of the encoded cards, and instead learned to select cards based on the first two bits (i.e., category). This is exactly what human participants are expected to distinguish when they are presented with the selection task; they automatically focus on card category as opposed to card instance.

As shown in Fig. 3, network 1 required three hidden units to learn the task. This was because pilot simulations revealed that three hidden units was the minimum number of hidden units required for the network to converge (i.e., to learn the desired mapping between input pattern and output response). If fewer than three hidden units were used, then the network was unable to generate the desired response to every pattern in the training set. We used the minimum number of hidden units to study the network's solution of the problem (and did not use more hidden units) because it has been argued that this kind of network is most likely to produce an internal structure that can be tractably interpreted (e.g., Berkeley et al., 1995).

Fig. 3 also shows that the network had four output units, one corresponding to each card. The network was trained to respond by turning 'on' one of its four output units. For example, output unit 1 turned 'on' (i.e., was activated to a value of 1) if card 1 held the desired response (i.e., affirmed the antecedent of the rule), but turned 'off' if card 1 did not hold this response. Only one output unit was activated in response to an input pattern because, for network 1, the desired response involved only the selection of the 'p' card. After the network learned the solution, we examined this 'mature' network for how it solved the task.

A mature network responds accurately and reliably to a complete set of training patterns. In this study, reliability of response required that (a) the network be able to identify the presented rule, (b) the network have some representation that assigned each input symbol to a more abstract category (e.g., differentiating between 'vowel' and 'odd number'), and (c) the network have some representation of the 'content' of the presented rule, such that its output would indicate what could be done to test the validity of the rule. We believe that network performance consistent with these three requirements for such a large number of different patterns reflecting the selection task has developed sufficient abilities to be of psychological interest. In short, such a network could be used to

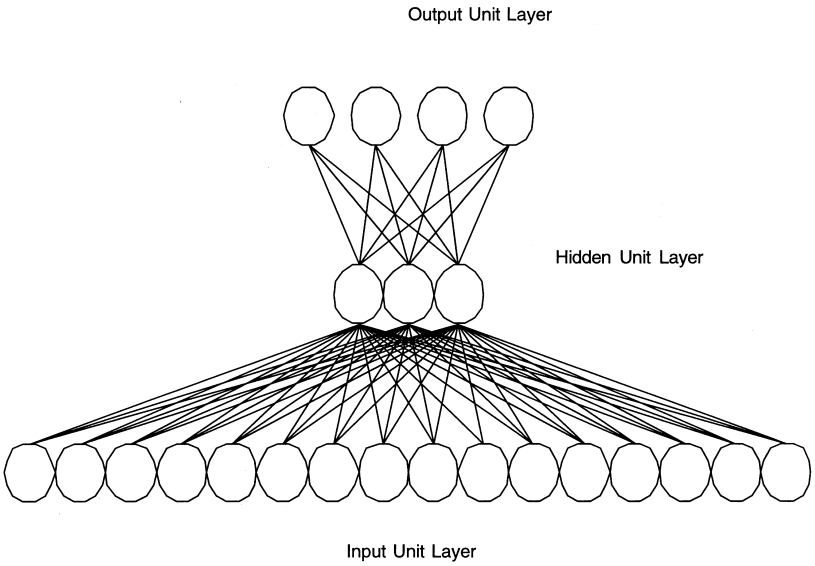

Fig. 3. Illustration of the PDP network trained to generate the card affirming the antecedent of the conditional rule (i.e., the 'p' card).

extend our understanding of selection task performance and existing algorithmic theories of the selection task.

3.1.2. Training

Network 1 was trained using Dawson and Schopflocher's (1992) modification of Rumelhart et al.'s (1986) generalized delta rule. The network was trained with a learning rate of 0.001 and a momentum of zero. Connection weights and unit biases (i.e., the mean of the Gaussian) were randomly selected from the range of -1.0 to $+1.0$. The weights and biases were updated after the presentation of each pattern. The order of pattern presentation was randomized during each epoch (an epoch is defined as the presentation of all patterns in the training set). This was done to ensure that the network's learning of the task was contingent on the specific input patterns and not on their specific sequence of presentation.

The network was trained until it generated a 'hit' in response to every pattern. A desired response or 'hit' consisted of an activation of 0.9 or higher in the output unit corresponding to the 'p' card along with activations of 0.1 or lower in output units corresponding to cards not affirming the antecedent of the rule. The network converged to a solution to this problem after 83 epochs. Following training, network performance on the selection task was considered comparable to human performance in so far as the network generated a reliable response to specific input. That is, the network generated the 'p' card in response to the rule in the task, which is consistent with a pattern typically shown by human participants.

3.2. Results: definite features of hidden units

The purpose of this first study was to interpret the method by which network 1 learned to solve the task; that is, learned to select the 'p' card in response to the conditional rule. Interpreting network 1 involved examining each of the network's hidden units for the input features it detected. Once the relationship between hidden units and input features was known, it was then possible to determine how the network solved the task.

The first step to understanding how the network solved the task involved *wire-tapping* each individual hidden unit for the input features it detected. Wire-tapping is one procedure that can be used to determine how PDP networks, in particular value unit networks, solve problems (e.g., Dawson, 1998; Moorehead, Haig, & Clement, 1989). Wiretapping involves recording the responses of the hidden units to the patterns in the training set. After the network is trained on a set of input patterns, the patterns are presented again to the network while their activities in individual hidden units are recorded. The recorded activities are plotted and examined for meaningful configurations. The configurations provide clues as to how the network is solving the task; that is, the configurations indicate which patterns fall into each of the decision regions created by the network to solve the task.

3.2.1. Jittered density plots

Jittered density plots of each hidden unit were constructed subsequent to wiretapping, as shown in Fig. 4. A jittered density plot illustrates the distribution of activation values produced in a single hidden unit of a mature network following a presentation of a full set of input patterns. A single dot in the plot represents the activation that one input pattern produces in a hidden unit. Hence, one plot illustrates as many dots as there are input patterns. The x-axis on the jittered density plot ranges from 0 to 1 and shows the range of activation values generated by the input pattern set. Dots are also randomly jittered along the y-axis to make them as discernible as possible.

Jittered density plots of value unit networks are frequently highly structured or 'banded.' The distinct bands represent groups of input patterns that share similar features and produce similar activations in a hidden unit. By examining the features that fall into each band, it is possible to identify the features that the network used to solve a problem. From the presence of the bands, we know that hidden units are reliably detecting specific input features in solving the task (Berkeley et al., 1995). Although bands provide information about the features used in solving a problem, the bands do not necessarily provide information about the problem's complexity. The number of hidden units is a better indicator of task complexity. While the number of hidden units

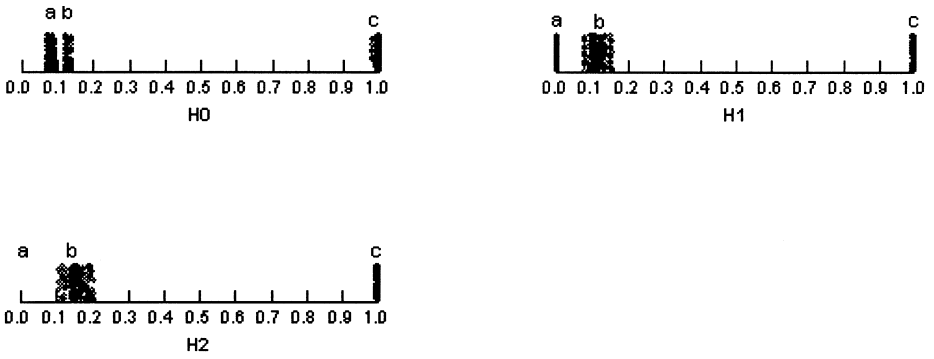

Fig. 4. Jittered density plots for each of the three hidden value units in network 1.

required by a network to solve a task indicates the number of decision regions in the pattern space, bands reflect the features that characterize the decision regions. As shown in Fig. 4, all three hidden units in network 1 exhibited a high degree of banding.

With the aid of descriptive statistics, it is possible to identify the specific features that cluster into each band and, moreover, how the network uses this configuration or 'carving of the input space' to solve a task. By calculating the Pearson product-moment correlation among the patterns in a band, it is possible to learn if a hidden unit is detecting definite binary features. A definite binary feature indicates a perfectly reliable correlation between input units. For example, a perfectly positive correlation between input units 6 and 7 indicates that these units always assume the same value; if input unit 6 has a value of 1.0, then so does input unit 7. In contrast, a perfectly negative correlation between a pair of input units suggests that whenever one input unit is 1.0, the other is 0.0 and vice versa. Network 1's hidden units detected only definite binary features. A description of the definite binary features detected by each hidden unit in network 1 is presented in Table 2.

Table 2
Definite features for bands from hidden units of network 1

Hidden unit	Band label	Definite features	n[a]
0	A	I0 ≠ I2, I0 ≠ I10, I2 = I10	1152
0	B	I0 ≠ I2, I1 ≠ I11	1152
0	C	I0 ≠ I2, I0 = I10, I1 = I11, I2 ≠ I10	768
1	A	I0 ≠ I2, I0 ≠ I7, I2 = I7	1152
1	B	I0 ≠ I2, I1 ≠ I8	1152
1	C	I0 ≠ I2, I0 = I7, I1 = I8 I2 ≠ I7	768
2	A	I0 ≠ I2, I0 ≠ I4, I2 = I4	1152
2	B	I0 ≠ I2, I1 ≠ I5	1152
2	C	I0 ≠ I2, I0 = I4, I1 = I5 I2 ≠ I4	768

' ≠ ' indicates a perfectly negative correlation between input units; ' = ' indicates a perfectly positive correlation between input units.

[a] n, number of patterns falling in each band.

3.2.2. Interpretation of definite binary features

An inspection of Table 2 suggests that network 1 solved the task by detecting binary features representing rules and specific cards. First, although each hidden unit detected all eight rules, each hidden unit detected a specific card. For example, notice that hidden unit 0 was highly activated by patterns (i.e., band C in Table 2) whose definite features showed input units 10 and 11 sharing a correlation of 1.0 with input units 0 and 1, respectively. Recall that input units 10 and 11 represent card 3 values, whereas input units 0 and 1 represent the antecedent of the rule. In other words, hidden unit 0 was highly activated when the desired response, 'p', was located at card 3. Notice also that hidden unit 0 was moderately activated by patterns (i.e., bands A and B) whose definite features showed input units 10 and 11 sharing a correlation of -1.0 with input units 0 and 1, respectively. In other words, hidden unit 0 was not highly activated when the value at card 3 failed to match the antecedent of the rule.

The same analysis can be applied to hidden units 1 and 2. For example, hidden unit 1 detected desired

responses at card 2 (i.e., inputs units 7 through 9), and hidden unit 2 detected desired responses at card 1 (i.e., input units 4 through 6). Although card 4 values were not directly detected by any of the hidden units, card 4 values were detected indirectly by all three hidden units. In value unit architectures, it is possible for a zero signal to moderate high activity or a '1' response if both the 'bias' of the architecture is equal to zero and the signal being sent is equal to zero. In other words, in a state where none of the three hidden units were activated (i.e., the desired response was not found at card 1, 2 or 3), this state activated output unit or card 4 (i.e., input units 13 through 15).

3.3. Discussion

Network 1 learned to select the 'p' card in response to the conditional rule by having each hidden unit detect values at specific card locations. This 'specialized' hidden unit algorithm did not discriminate among the rules used in the task since all three hidden units detected all rules. Instead, the specialized algorithm discriminated among card locations to solve the task. We speculate that the network's specific focus on card location points to the importance of the evidence in the selection task. The cards in the selection task illustrate the evidence with which to test the rule. Although few existing theories of selection task performance focus on the evidence, some investigators have proposed that the kind of evidence available to participants plays an important role in how participants choose to test a rule (e.g., Klayman & Ha, 1987; Liberman & Klar, 1996). For example, Liberman & Klar (1996) suggested that if participants perceive the evidence needed to test a hypothesis as atypical, they might forego using the atypical evidence and choose to test the hypothesis using more conventional evidence. The results obtained from network 1 suggest that the evidence in the selection task might play an important role in participants' responses.

At this time, it is unclear how the findings obtained from network 1 support or challenge existing theories of selection task performance. For example, Cheng and Holyoak's (1985, 1989) pragmatic reasoning theory is vague about how participants might view the evidence or its role in reasoning generally. Cosmides' (1989) social contract theory and Johnson-Laird's mental models theory (1983) are also unclear about the influence the evidence might have on participants' performance. All three theories focus primarily on the nature of the rule and its associated context. In contrast, according to Rips' (1994) syntactic theory, the nature of the evidence should have little influence on performance since mental rules operate on the syntax of the selection task rule. In the next study, we trained a network to solve the task by selecting the 'p' and 'not-q' cards. This is a response rarely exhibited by human participants.

4. Network 2: selection of the 'p' card and the 'not-q' card

4.1. Method

4.1.1. Network architecture

Network 2 was trained under the same method used to train network 1 — identical input encoding, training patterns, and output units were used. This time, however, two output units instead of one were designed to turn 'on' in response to every input pattern since the solution of the task involved the selection of two 'cards'. As shown in Fig. 5, network 2 required eight hidden units instead of three to learn the task. Pilot simulations revealed that the network could not learn to generate the desired mapping from inputs to outputs with fewer than eight hidden units when the network was required to select two 'cards.'

4.1.2. Training

Network 2 was trained similarly to network 1 with the exception that network 2 was trained to select two responses instead of one. Because network 2 selected two responses, it needed to learn to distinguish between propositions — 'p' and 'not-q' — in its responses. We did not indicate to the network before its training which values represented 'p' and which values represented 'not-q' because learning to make this distinction is an integral part of learning the task. When human beings solve the selection task, they approach the task already knowing which

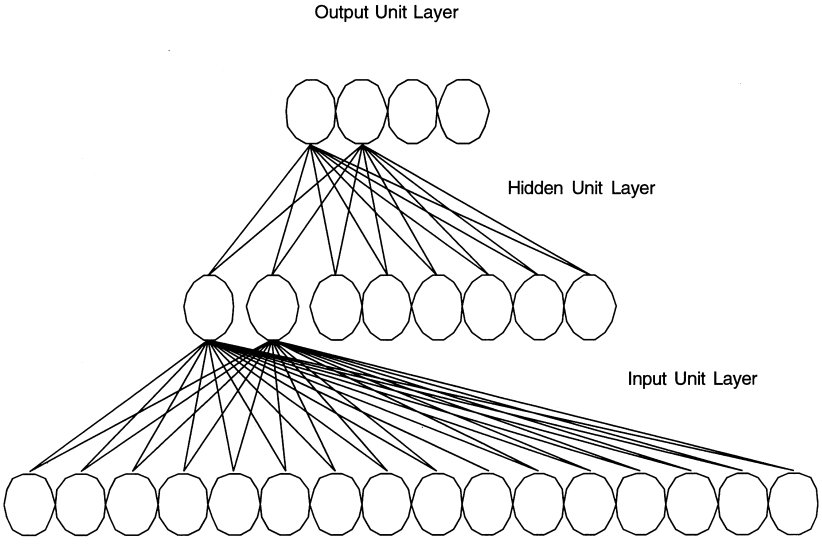

Fig. 5. Illustration of the PDP network trained to generate both the card affirming the antecedent and the card negating the consequent of the conditional rule (i.e., the 'p' and the 'not-q' cards).

values represent 'p' and which values represent 'not-q,' but at some point in their learning history human beings have had to learn to make the distinction between propositions. Learning to distinguish propositions was part of the network's training on the task.

As with network 1, prior to training, the network's connection weights were randomly set to values between -1.0 and $+1.0$, while unit biases were set to 0.1. The learning rate was 0.001 and no momentum was used. At the end of training, the network generated a 'hit' in response to every pattern. A desired response or 'hit' consisted of an activation of 0.9 or higher in the output units corresponding to the desired response (i.e., 'p' or 'not-q'). An activation of 0.1 or lower characterized outputs units corresponding to other responses. The network learned to generate the desired response to all patterns after 414 epochs.

In addition, because the network was examined after it was presented a large number of training patterns, it had the opportunity to determine for itself the underlying nature of the task. For example, all of the networks that we trained learned to ignore the last 'bit' of the encoded cards, and instead learned to select cards based on the first two bits (i.e., category). This is exactly what human participants are expected to distinguish when they are presented with the selection task; they automatically focus on card category as opposed to card instance.

4.1.3. Results

Network 2 was analyzed following the same approach used to analyze network 1. As illustrated in Figs. 6 and 7, jittered density plots of each of the eight hidden units in network 2 revealed a high degree of banding. Such pronounced banding suggested that the network's solution to the task involved the detection of definite features. An examination of the plots, furthermore, suggested that pairs of hidden units had similar patterns of banding; for example, hidden units 0 and 6, hidden units 1 and 4, hidden units 2 and 5, and hidden units 3 and 7. The visual similarity observed in the plots between pairs of hidden units was supported when we ran a correlation among hidden unit activity. In particular, we discovered that the activity of hidden units 3 and 7 shared a correlation of -0.99 when a desired response was located at card 1 (see Tables 3 and 4); hidden units 0 and 6 shared a correlation of 0.99 when a desired response was located at card 2; hidden units 2 and 5 shared a correlation of 1 when a desired response was located at card 3; and hidden units 1 and 4 shared a correlation of 0.81 when a desired response was located at card 4. Strong

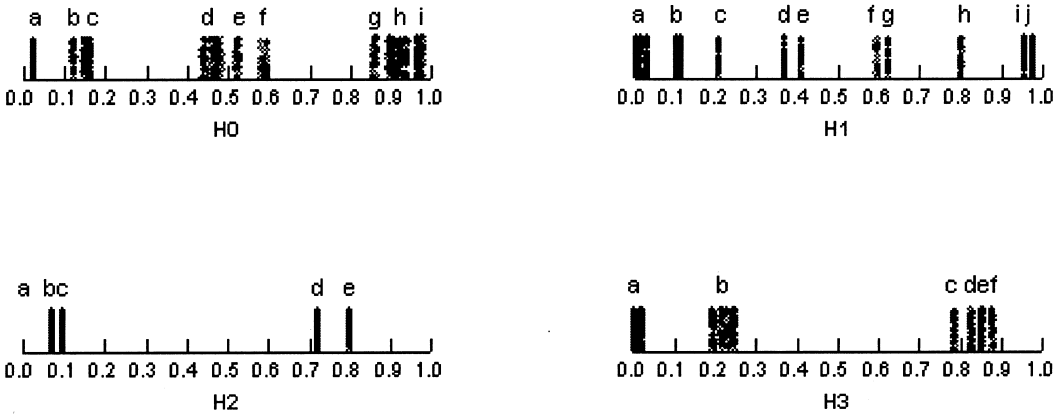

Fig. 6. Jittered density plots for the first set of four hidden value units used in network 2.

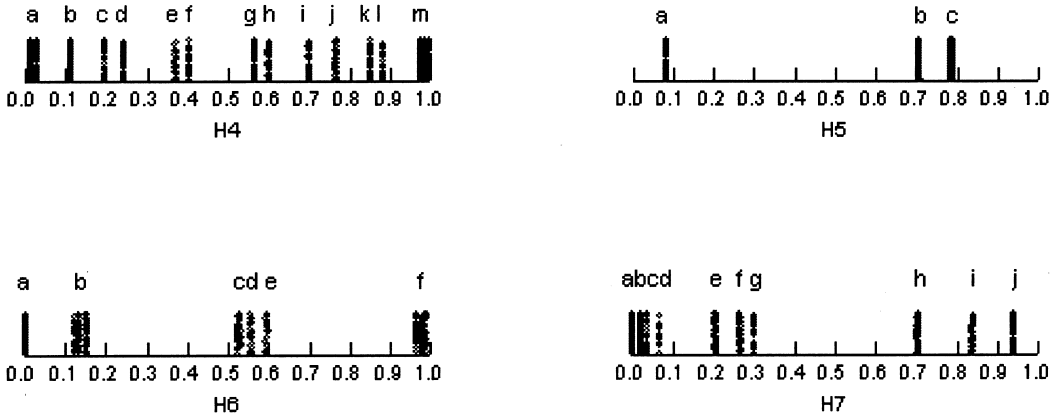

Fig. 7. Jittered density plots for the second set of four hidden value units used in network 2.

correlations between pairs of hidden units disappeared when undesired responses were located at their respective card locations.

An examination of the definite features detected by network 2's hidden units suggested a 'specialized' algorithm similar to that found for network 1. Tables 5–8 show the exhaustive list of definite features that all eight hidden units detected. To illustrate network 2's algorithm, we will focus on hidden unit 2 (see Table 7). Although it would be too laborious to describe here the entire list of definite features that hidden unit 2 detected, decoding the list indicates that hidden unit 2 was highly activated by patterns whose definite features had the following values at input units 1 through 4: 0011, 1100, 0110, 1001, 0111, 1110, 0010, 1101. Recall that input units 1 through 4 represent the rule of the task, the first two bits represent the antecedent of the rule while the last two bits represent the consequent. Hidden unit 2 was also highly activated by patterns that had a desired response (in relation to the rule) located at card 3. This means that input patterns that had a desired response — 'p' or 'not-q' — located at card 3 highly activated hidden unit 2.

Hidden unit 2's activity was highly correlated with hidden unit 5's activity in detecting desired responses at card 3. Decoding the list of definite features detected by hidden unit 5 indicates that this hidden unit was highly activated by patterns that had the following values at input units 1 through 4: 0011, 1100, 0110, 1001, 0111, and 1000. Hidden unit 5 did not detect any definite features associated with card

Table 3
Correlation among hidden units when cards 1 and 2 are selected — network 2

	H0	H1	H2	H3	H4	H5	H6	H7
Card 1:								
H0	1							
H1	0.09	1						
H2	0.16	− 0.32	1					
H3	0	− 0.11	0	1				
H4	0.02	− 0.31	0.02	− 0.19	1			
H5	0	− 0.24	− 0.28	0.154	0.18	1		
H6	0.542	− 0.1	− 0.1	− 0.13	0.04	0.16	1	
H7	0	0.09	0	**− 0.99**	0.212	0	0.144	1
Card 2:								
H0	1							
H1	0	1						
H2	0.04	− 0.32	1					
H3	− 0.16	− 0.25	− 0.21	1				
H4	0.06	− 0.31	0.02	0.03	1			
H5	− 0.18	− 0.24	− 0.28	0.02	0.181	1		
H6	**0.996**	0	0.02	− 0.16	0.07	− 0.1	1	
H7	0.191	− 0.11	− 0.13	− 0.34	0.197	0.105	0.21	1

Table 4
Correlation among hidden units when cards 3 and 4 are selected — network 2

	H0	H1	H2	H3	H4	H5	H6	H7
Card 3:								
H0	1							
H1	0.09	1						
H2	0	− 0.24	1					
H3	0.307	− 0.25	0.03	1				
H4	0.02	− 0.31	0.18	0.03	1			
H5	0	− 0.24	**1**	0.03	0.18	1		
H6	0.541	− 0.1	0.161	0.03	0.04	0.16	1	
H7	− 0.1	− 0.11	0.106	− 0.34	0.196	0.105	− 0.22	1
Card 4:								
H0	1							
H1	− 0.1	1						
H2	0.16	− 0.21	1					
H3	0.309	− 0.1	− 0.21	1				
H4	− 0.1	**0.814**	− 0.1	− 0.15	1			
H5	0	0.748	− 0.28	0.03	0.224	1		
H6	0.541	0.07	− 0.1	0.03	0	0.159	1	
H7	− 0.1	0.173	− 0.13	− 0.34	0.163	0.106	− 0.22	1

values. In short, hidden units 2 and 5 were both highly activated by a large set of rules and helped to detect responses located at card 3. Similar accounts may be made for the remaining pairs of hidden units (see Tables 5, 6 and 8).

4.2. Discussion

Network 2 solved the task by means of specialized ‘pairs’ of hidden units. In particular, hidden units 0 and 6 detected desired responses located at card 2,

Table 5
Definite features for bands from hidden units 0 and 6 of network 2

H0-bands	Definite features	n[a]	H6-bands	Definite features	n
A	I0 ≠ I1, I0 ≠ I2, I0 = I3, I0 ≠ I7, I0 ≠ I8, I1 = I2, I1 ≠ I3, I1 = I7, I1 = I8, I2 ≠ I3, I2 = I7, I2 = I8, I3 ≠ I7, I3 ≠ I8, I7 = I8	192	A	I0 ≠ I2	1344
			B	I0 ≠ I2, I1 ≠ I3	576
			C	I0 = I1, I0 ≠ I2, I0 ≠ I3, I0 = I7, I0 ≠ I8, I1 ≠ I2, I1 ≠ I3, I1 = I7, I1 ≠ I8, I2 = I3, I2 ≠ I7, I2 = I8, I3 ≠ I7, I3 = I8, I7 ≠ I8	192
B	I0 = I1, I0 ≠ I2, I0 = I3, I0 ≠ I7, I0 ≠ I8, I1 ≠ I2, I1 = I3, I1 ≠ I7, I1 ≠ I8, I2 ≠ I3, I2 = I7, I2 = I8, I3 ≠ I7, I3 ≠ I8, I7 = I8	192			
			D	I0 = I1, I0 ≠ I2, I0 ≠ I3, I0 ≠ I7, I0 = I8, I1 ≠ I2, I1 ≠ I3, I1 ≠ I7, I1 = I8, I2 = I3, I2 = I7, I2 ≠ I8, I3 = I7, I3 ≠ I8, I7 ≠ I8	192
C	I0 ≠ I2, I0 = I7, I1 ≠ I3 I1 = I8, I2 ≠ I7, I3 ≠ I8	384			
D	I0 ≠ I2, I1 = I3	576			
E	I0 = I1, I0 ≠ I2, I0 ≠ I3, I0 ≠ I7, I0 = I8, I1 ≠ I2, I1 ≠ I3, I1 ≠ I7, I1 = I8, I2 = I3, I2 = I7, I2 ≠ I8, I3 = I7, I3 ≠ I8, I7 ≠ I8	192	E	I0 ≠ I1, I0 ≠ I2, I0 = I3, I0 = I7, I0 = I8, I1 = I2, I1 ≠ I3, I1 ≠ I7, I1 ≠ I8, I2 ≠ I3, I2 ≠ I7, I2 ≠ I8 I3 = I7, I3 = I8, I7 = I8	192
F	I0 ≠ I1, I0 ≠ I2, I0 = I3, I0 ≠ I7, I0 = I8, I1 = I2, I1 ≠ I3, I1 = I7, I1 ≠ I8, I2 ≠ I3, I2 = I7, I2 ≠ I8, I3 ≠ I7, I3 = I8, I7 ≠ I8	192			
G	I0 = I1, I0 ≠ I2, I0 ≠ I3, I0 ≠ I7, I0 ≠ I8, I1 ≠ I2, I1 ≠ I3, I1 ≠ I7, I1 ≠ I8, I2 = I3, I2 = I7, I2 = I8, I3 = I7, I3 = I8, I7 = I8	192			
H	I0 ≠ I2, I1 = I3	768			
I	I0 ≠ I2, I0 = I7, I1 ≠ I3, I1 ≠ I8, I2 ≠ I7, I3 = I8	384			

' ≠ ' indicates a perfectly negative correlation between input units; ' = ' indicates a perfectly positive correlation between input units.
[a] n, number of patterns falling in each band.

hidden units 1 and 4 detected desired responses located at card 4, hidden units 2 and 5 detected desired responses located at card 3, and hidden units 3 and 7 detected desired responses located at card 1.

In comparison to the task network 1 had to solve, network 2's task was more difficult. Whereas network 1 required only three hidden units to solve its task, network 2 required eight hidden units to solve its task. It is not surprising that network 2 required a greater number of hidden units to solve its task, however, given that it generated two responses instead of just the one. Nevertheless, what is striking about both network 1 and network 2 is that their algorithms for solving the tasks involved 'specialized' hidden units detecting desired responses at specific card locations.

5. Network 3: selection of the 'p' card and the 'q' card

Both networks 1 and 2 solved the task by means of specialized hidden units that focused on card location. That hidden units specialized to discrimi-

nate card locations was intriguing to us because this aspect of the selection task has not been examined closely in the past. However, we were not only intrigued by this finding but also by the apparent difficulty of generating two card values instead of just one. Although we expected the selection of two card values to be a more difficult task, we did not expect network 2 to need more than double the hidden units required to train network 1. Recall that the number of hidden units required by a network to solve a task is indicative of the task's difficulty. We conjectured that another reason for network 2's task difficulty might have something to do with needing to select the 'not-q' response. Solving the task by selecting the 'not-q' response is highly uncommon for human participants; it is much more common for participants to select the 'p' card or both the 'p' and 'q' cards (Evans et al., 1993).

We tested our conjecture by training a third network to select both the 'p' and the 'q' cards. In this way, we hold constant the number of cards the network must select in responding to the task, while at the same time testing to see if selecting the 'q' response is easier to learn than the 'not-q' response. Training a third network to select the 'p' and 'q' also served as a further test of the algorithms found for both networks 1 and 2.

5.1. Method

5.1.1. Network architecture

The same method used to train both networks 1 and 2 was used to train network 3. As with network 2, two output units were designed to turn 'on' in response to every input pattern since the desired response involved the selection of two cards. Surprisingly, we found that network 3 also required eight hidden units to converge, which is the same number of hidden units required by network 2. We had expected network 3 to require fewer hidden units based on our prediction that selecting the 'q' card might be more easily learned than selecting the 'not-q' card.

5.1.2. Training

As with networks 1 and 2, prior to training, network 3's connection weights were randomly set to values between -1.0 to $+1.0$, while its unit biases were set to zero. The learning rate was 0.001 and no momentum was used. At the end of training, the network generated a 'hit' in response to every pattern. A desired response or 'hit' consisted of an activation of 0.9 or higher in the output units corresponding to the 'p' and 'q' cards, and an activation of 0.1 or lower in output units corresponding to the other cards. The network learned to select the desired responses after 115 epochs of training.

5.2. Results

As shown in Figs. 8 and 9, jittered density plots of each of the hidden units displayed a high degree of banding, which suggested that network 3's solution involved the detection of definite features. An inspection of the jittered density plots indicated that some hidden units were detecting similar definite features. For example, four out of the eight hidden units (i.e., hidden units 0, 2, 4, and 6) had similar jittered density plots. Tables 9 and 10 show the correlations among the hidden unit activity levels. From the tables, we see that the activity levels of both hidden units 3 and 4 and hidden units 5 and 7 correlated 0.99 when a desired response was located at card 1. In contrast, when a desired response was located at card 2, hidden units 0 and 4 shared a correlation of 0.99. Moreover, when a desired response was located at card 3, hidden units 4 and 6 shared a correlation of 0.99, and when a desired response was located at card 4, hidden unit 2 and 4 shared a correlation of 0.99. Unlike network 2, detecting a desired response at card 1 required four hidden units, while detecting desired responses at other card locations required only two. In addition, hidden unit 4 was involved in all response selections.

A closer analysis of how network 3 solved the task revealed several interesting results. First, the hidden units in network 3 did not detect as many definite features in solving the task as we found for network 2. For example, notice that network 3's jittered density plots did not exhibit as many individual bands for each hidden unit as we observed for network 2. Second, and more specifically, an inspection of the definite features detected by hidden unit 4 revealed a correspondence to the four input units used to represent the rule in the task. Table 13 shows

Table 6
Definite features for bands from hidden units 1 and 4 of network 2

H1-bands	Definite features	n[a]	H4-bands	Definite features	n
A	I0 ≠ I2, I1 ≠ I14	960	A	I0 = I1, I0 ≠ I2, I0 ≠ I13, I0 = I14, I1 ≠ I2, I1 ≠ I13, I1 = I14, I2 = I13, I2 ≠ I14, I13 ≠ I14	384
B	I0 ≠ I1, I0 ≠ I2, I0 ≠ I3, I1 = I2, I1 = I3, I2 = I3, I13 = I14	384			
C	I0 = I1, I0 ≠ I2, I0 = I3, I0 = I13, I0 ≠ I14, I1 ≠ I2, I1 = I3, I1 ≠ I13, I1 ≠ I4, I2 ≠ I3, I2 ≠ I13, I2 = I14, I3 = I13, I3 ≠ I14, I13 ≠ I14	192	B	I0 = I1, I0 ≠ I2, I0 = I3, I0 ≠ I13, I0 ≠ I14, I1 ≠ I2, I1 = I3, I1 ≠ I13, I1 ≠ I14, I2 ≠ I3, I2 = I13, I2 = I14, I3 ≠ I13, I3 ≠ I14, I13 = I14	192
D	I0 = I1, I0 ≠ I2, I0 = I3, I0 ≠ I13, I0 = I14, I1 ≠ I2, I1 ≠ I3, I1 ≠ I13, I1 = I14, I2 = I3, I2 = I13, I2 ≠ I14, I3 = I13, I3 ≠ I14, I13 ≠ I14	192	C	I0 = I1, I0 ≠ I2, I0 ≠ I3, I0 ≠ I13, I0 ≠ I14, I1 ≠ I2, I1 ≠ I3, I1 ≠ I13, I1 ≠ I14, I2 = I3, I2 = I13, I2 = I14, I3 = I13, I3 = I14, I13 = I14	192
E	I0 = I1, I0 ≠ I2, I0 = I3, I0 = I13, I0 = I14, I1 ≠ I2, I1 = I3, I1 = I13, I1 = I14, I2 ≠ I3, I2 ≠ I13, I2 ≠ I14, I3 = I13, I3 = I14, I13 = I14	192	D	I0 ≠ I1, I0 ≠ I2, I0 = I3, I0 ≠ I13, I0 = I14, I1 = I2, I1 ≠ I3, I1 = I13, I1 ≠ I14, I2 ≠ I3, I2 = I13, I2 ≠ I14, I3 ≠ I13, I3 = I14, I13 ≠ I14	192
F	I0 ≠ I1, I0 ≠ I2, I0 ≠ I3, I0 ≠ I13, I0 = I14, I1 = I2, I1 = I3, I1 = I13, I1 ≠ I14, I2 = I3, I2 = I13, I2 = I14, I3 = I13, I3 ≠ I14, I13 ≠ I14	192	E	I0 ≠ I1, I0 ≠ I2, I0 ≠ I3, I0 ≠ I13, I0 = I14, I1 = I2, I1 = I3, I1 = I13, I1 ≠ I14, I2 = I3, I2 = I13, I2 ≠ I14, I3 = I13, I3 ≠ I14, I13 ≠ I14	192
G	I0 ≠ I1, I0 ≠ I2, I0 ≠ I3, I0 = I13, I0 ≠ I14, I1 = I2, I1 = I3, I1 ≠ I13, I1 = I14, I2 = I3, I2 ≠ I13, I2 = I14, I3 ≠ I13, I3 = I14, I13 ≠ I14	192	F	I0 ≠ I1, I0 ≠ I2, I0 ≠ I3, I0 = I13, I0 ≠ I14, I1 = I2, I1 = I3, I1 ≠ I13, I1 = I14, I2 = I3, I2 ≠ I13, I2 = I14, I3 ≠ I13, I3 = I14, I13 ≠ I14	192
H	I0 ≠ I1, I0 ≠ I2, I0 = I3, I0 = I13, I0 ≠ I14, I1 = I2, I1 ≠ I3, I1 ≠ I13, I1 = I14, I2 ≠ I3, I2 ≠ I13, I2 = I14, I3 = I13, I3 ≠ I14, I13 ≠ I14	192	G	I0 ≠ I1, I0 ≠ I2, I0 = I3, I0 = I13, I0 ≠ I14, I1 = I2, I1 ≠ I3, I1 ≠ I13, I1 = I14, I2 ≠ I3, I2 ≠ I13, I2 = I14, I3 = I13, I3 ≠ I14, I13 ≠ I14	192
I	I0 ≠ I1, I0 ≠ I2, I0 = I3, I0 ≠ I13, I0 ≠ I14, I1 = I2, I1 ≠ I3, I1 = I13, I1 = I14, I2 ≠ I3, I2 = I13, I2 = I14, I3 ≠ I13, I3 ≠ I14, I13 = I14	192	H	I0 = I1, I0 ≠ I2, I0 = I3, I0 = I13, I0 = I14, I1 ≠ I2, I1 = I3, I1 = I13, I1 = I14, I2 ≠ I3, I2 ≠ I13, I2 ≠ I14, I3 = I13, I3 = I14, I13 = I14	192
J	I0 = I1, I0 ≠ I2, I0 ≠ I3, I0 = I13, I0 = I14, I1 ≠ I2, I1 ≠ I3, I1 = I13, I1 = I14, I2 = I3, I2 ≠ I13, I2 ≠ I14, I3 ≠ I13, I3 ≠ I14, I13 = I14	192	I	I0 ≠ I1, I0 ≠ I2, I0 = I3, I0 ≠ I13, I0 ≠ I14, I1 = I2, I1 ≠ I3, I1 = I13, I1 = I14, I2 ≠ I3, I2 = I13, I2 = I14, I3 ≠ I13, I3 ≠ I14, I13 = I14	192
K	I0 = I1, I0 ≠ I2, I0 = I3, I0 ≠ I13, I0 = I14, I1 ≠ I2, I1 = I3, I1 ≠ I13, I1 = I14, I2 ≠ I3, I2 = I13, I2 ≠ I14, I3 ≠ I13, I3 = I14, I13 ≠ I14	192	J	I0 = I1, I0 ≠ I2, I0 ≠ I3, I0 = I13, I0 = I14, I1 ≠ I2, I1 ≠ I3, I1 = I13, I1 = I14, I2 = I3, I2 ≠ I13, I2 ≠ I14, I3 ≠ I13, I3 ≠ I14, I13 = I14	192

Table 6. Continued

H1-bands	Definite features	n[a]	H4-bands	Definite features	n
			K	I0 ≠ I1, I0 ≠ I2, I0 ≠ I3, I0 ≠ I13, I0 ≠ I14, I1 = I2, I1 = I3, I1 = I13, I1 = I14, I2 = I3, I2 = I13, I2 = I14, I3 = I13, I3 = I14, I13 = I14	192
			L	I0 ≠ I1, I0 ≠ I2, I0 ≠ I3, I0 = I13, I0 = I14, I1 = I2, I1 = I3, I1 ≠ I13, I1 ≠ I14, I2 = I3, I2 ≠ I13, I2 ≠ I14, I3 ≠ I13, I3 ≠ I14, I13 = I14	192
			M	I0 ≠ I2, I0 = I3, I1 ≠ I14, I2 ≠ I13	576

‘≠’ indicates a perfectly negative correlation between input units; ‘=’ indicates a perfectly positive correlation between input units.
[a] n, number of patterns falling in each band.

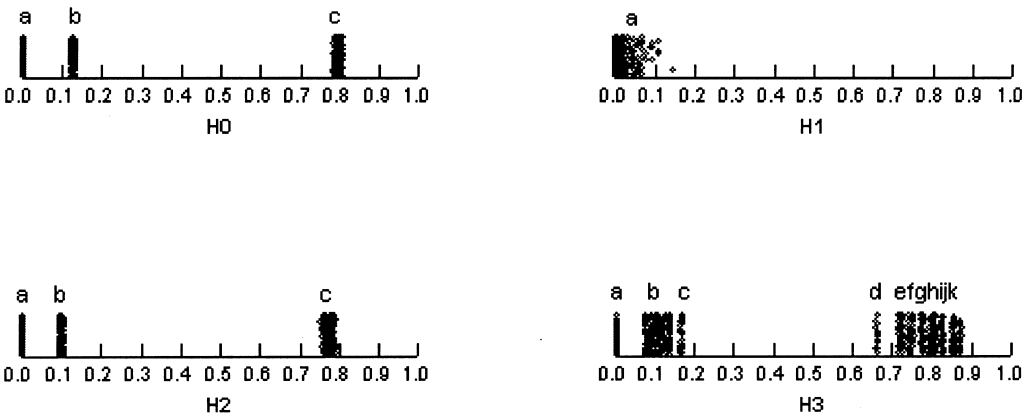

Fig. 8. Jittered density plots for the first set of four hidden value units used in network 3.

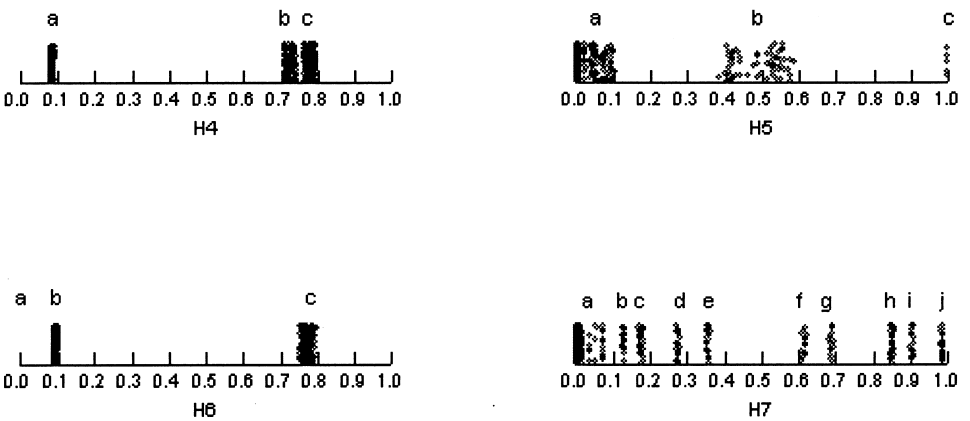

Fig. 9. Jittered density plots for the second set of four hidden value units used in network 3.

Table 7
Definite features for bands from hidden units 2 and 5 of network 2

H2-bands	Definite features	n[a]	H5-bands	Definite features	n
A	I0 ≠ I2, I1 ≠ I3, I1 ≠ I10, I1 ≠ I11, I3 = I10, I3 = I11, 10 = I11	384	A	I0 = I1, I0 ≠ I2, I0 = I3, I1 ≠ I2, I1 = I3, I2 ≠ I3	768
B	I0 ≠ I2, I1 ≠ I3, I1 = I10, I1 ≠ I11, I3 ≠ I10, I3 = I11, I10 ≠ I11	384	B	I0 ≠ I1, I0 ≠ I2, I0 ≠ I3, I1 = I2, I1 = I3, I2 = I3	768
C	I0 ≠ I2, I1 = I3, I10 = I11	768	C	I0 ≠ I2, I1 ≠ I3	1536
D	I0 ≠ I2, I1 = I3, I10 ≠ I11	768			
E	I0 ≠ I2, I1 ≠ I3, I1 = I11, I3 ≠ I11	768			

‘≠’ indicates a perfectly negative correlation between input units; ‘=’ indicates a perfectly positive correlation between input units.
[a] *n*, number of patterns falling in each band.

the definite features that activated hidden unit 4. After decoding the set of definite features, we see that hidden unit 4 detected all the rules used to train the network: 0011, 1100, 0110, 1001, 0010, 1101, 0111, and 1000. Hidden unit 4, however, failed to detect any definite features associated with card location. Hidden unit 4 in network 3 was similar to network 2’s hidden unit 5, except that hidden unit 4 enabled all remaining hidden units to detect desired responses (i.e., hidden unit 4 correlates highly with all hidden units when desired responses are located at corresponding card locations) whereas network 2’s hidden unit 5 only enabled hidden unit 2 to detect desired responses.

An inspection of the remaining hidden units revealed that network 3’s algorithm for solving the task was similar to network 2’s algorithm but with some differences. For example, Table 11 shows the definite features that activated hidden unit 0. After decoding the definite features, we see that hidden unit 0 was highly to moderately activated by patterns that had the following values at input units 1 through 4: 0011, 1100, 0111, 1000, 0010, 1101, 0110, and 1001. In addition, hidden unit 0 was activated by responses located at card 2. Hidden unit 0 along with hidden unit 4 helped to detect desired responses located at card 2; hidden unit 0 detected rules and responses at card 2, whereas hidden unit 4 only detected rules. This division of labor among hidden units in network 3 is slightly different from that found in network 2, in which pairs of hidden units each detected rules and responses at specific card locations. This analysis can also be used to understand the collaboration of hidden units in activating outputs 1, 3 and 4 (see Tables 11–13).

5.3. Discussion

We trained network 3 in order to explore the difficulty associated with network 2’s task. We wanted to find out whether the task difficulty of selecting both the ‘p’ and ‘not-q’ cards originated from network 2’s need to select two cards or from the rigor of learning the ‘not-q’ selection. We attempted to answer this question by training network 3 to solve the task by selecting both the ‘p’ and the ‘q’ cards. In this way we held constant the number of selections made by the network, while at the same time testing to see if selecting the ‘q’ response required fewer hidden units than network 2. Training network 3 also served as a further test of the specialized algorithm found for both networks 1 and 2.

Our results indicated that selecting the ‘p’ and ‘q’ response to the selection task is as difficult to generate as selecting the ‘p’ and ‘not-q’ response. Network 3 required eight hidden units to learn to solve the task — the same number of hidden units as network 2 required. Although network 3 required fewer epochs to converge to a solution (i.e., 115 for

Table 8
Definite features for bands from hidden units 3 and 7 of network 2

H3-bands	Definite features	n[a]	H7-bands	Definite features	n
A	I0 ≠ I2	1536	A	I0 ≠ I2	1152
B	I0 ≠ I2	768	B	I0 = I1, I0 ≠ I2, I0 = I3, I0 = I4, I0 ≠ I5, I1 ≠ I2, I1 = I3, I1 = I4, I1 ≠ I5, I2 ≠ I3, I2 ≠ I4, I2 = I5, I3 = I4, I3 ≠ I5, I4 ≠ I5	192
C	I0 = I1, I0 ≠ I2, I0 = I3, I0 = I4, I0 = I5, I1 ≠ I2, I1 = I3, I1 = I4, I1 = I5, I2 ≠ I3, I2 ≠ I4, I2 ≠ I5, I3 = I4, I3 = I5, I4 = I5	192	C	I0 ≠ I1, I0 ≠ I2, I0 = I3, I0 ≠ I4, I0 ≠ I5, I1 = I2, I1 ≠ I3, I1 = I4, I1 = I5, I2 ≠ I3, I2 = I4, I2 = I5, I3 ≠ I4, I3 ≠ I5, I4 = I5	192
D	I0 = I1, I0 ≠ I2, I0 ≠ I3, I0 ≠ I4, I0 = I5, I1 ≠ I2, I1 ≠ I3, I1 ≠ I4, I1 = I5, I2 = I3, I2 = I4, I2 ≠ I5, I3 = I4, I3 ≠ I5, I4 ≠ I5	192	D	I0 = I1, I0 ≠ I2, I0 ≠ I3, I0 ≠ I4, I0 = I5, I1 ≠ I2, I1 ≠ I3, I1 ≠ I4, I1 = I5, I2 = I3, I2 = I4, I2 ≠ I5, I3 = I4, I3 ≠ I5, I4 ≠ I5	192
E	I0 ≠ I1, I0 ≠ I2, I0 = I3, I0 ≠ I4, I0 ≠ I5, I1 = I2, I1 ≠ I3, I1 = I4, I1 = I5, I2 ≠ I3, I2 = I4, I2 = I5, I3 ≠ I4, I3 ≠ I5, I4 = I5	192	E	I0 = I1, I0 ≠ I2, I0 ≠ I3, I0 ≠ I4, I0 ≠ I5, I1 ≠ I2, I1 ≠ I3, I1 ≠ I4, I1 ≠ I5, I2 = I3, I2 = I4, I2 = I5, I3 = I4, I3 = I5, I4 = I5	192
F	I0 = I1, I0 ≠ I2, I0 ≠ I3, I0 = I4, I0 = I5, I1 ≠ I2, I1 ≠ I3, I1 = I4, I1 = I5, I2 = I3, I2 ≠ I4, I2 ≠ I5, I3 ≠ I4, I3 ≠ I5, I4 = I5	192	F	I0 = I1, I0 ≠ I2, I0 ≠ I3, I0 = I4, I0 ≠ I5, I1 ≠ I2, I1 ≠ I3, I1 = I4, I1 ≠ I5, I2 = I3, I2 ≠ I4, I2 = I5, I3 ≠ I4, I3 = I5, I4 ≠ I5	192
			G	I0 ≠ I1, I0 ≠ I2, I0–I3, I0 ≠ 14, I0 = I5, I1 = I2, I1 ≠ I3, I1 = I4, I1 = I5, I2 ≠ I3, I2 ≠ I4, I2 = I5, I3 ≠ I4, I3 ≠ 15, I4 = I5	192
			H	I0 ≠ I1, I0 ≠ I2, I0 = I3, I0 = I4, I0 ≠ I5, I1 = I2, I1 ≠ I3, I1 ≠ I4, I1 = I5, I2 ≠ I3, I2 ≠ I4, I2 = I5, I3 = I4, I3 ≠ I5, I4 ≠ I5	192
			I	I0 = I1, I0 ≠ I2, I0 = I3, I0 ≠ I4, I0 ≠ I5, I1 ≠ I2, I1 = I3, I1 ≠ I4, I1 ≠ I5, I2 ≠ I3, I2 = I4, I2 = I5, I3 ≠ I4, I3 ≠ I5, I4 = I5	192
			J	I0 ≠ I1, I0 ≠ I2, I0 ≠ I3, I1 = I2, I1 = I3, I2 = I3, I4 ≠ I5	384

‘ ≠ ’ indicates a perfectly negative correlation between input units; ‘ = ’ indicates a perfectly positive correlation between input units.
[a] n, number of patterns falling in each band.

network 3 versus 414 for network 2), the number of epochs a network requires to learn a task is not an unequivocal indicator of a task’s difficulty. A relatively high number of epochs might only suggest that the network had a ‘bad start’ — that is, the network’s initial random weights were highly dissimilar from the network’s final weights. The number of hidden units required by a network to converge to a

Table 9
Correlation among hidden units when cards 1 and 2 are selected — network 3

	H0	H1	H2	H3	H4	H5	H6	H7
Card 1:								
H0	1							
H1	0	1						
H2	− 0.36	0	1					
H3	− 0.31	0.131	− 0.3	1				
H4	− 0.31	0.11	− 0.31	**0.987**	1			
H5	− 0.1	0	− 0.1	0.265	0.249	1		
H6	− 0.35	− 0.1	− 0.36	− 0.3	− 0.31	− 0.1	1	
H7	− 0.1	0	− 0.1	0.298	0.269	**0.988**	− 0.1	1
Card 2:								
H0	1							
H1	0.06	1						
H2	− 0.3	0	1					
H3	− 0.3	0.05	− 0.36	1				
H4	**0.997**	0.06	− 0.31	− 0.3	1			
H5	0.169	0.07	− 0.18	0.223	0.169	1		
H6	− 0.3	0	− 0.36	− 0.35	− 0.31	− 0.18	1	
H7	− 0.14	0.103	0.05	0.06	− 0.16	0.249	0.05	1

Table 10
Correlation among hidden units when cards 3 and 4 are selected — network 3

	H0	H1	H2	H3	H4	H5	H6	H7
Card 3:								
H0	1							
H1	0.02	1						
H2	− 0.36	0	1					
H3	− 0.35	0.08	− 0.36	1				
H4	− 0.31	− 0.1	− 0.31	− 0.3	1			
H5	− 0.18	− 0.14	− 0.18	0.222	0.168	1		
H6	− 0.31	− 0.1	− 0.31	− 0.3	**0.997**	0.164	1	
H7	0.05	0.09	0.05	0.06	− 0.16	0.247	− 0.14	1
Card 4:								
H0	1							
H1	0	1						
H2	− 0.31	0.02	1					
H3	− 0.36	0.07	− 0.3	1				
H4	− 0.31	0.02	**0.997**	− 0.3	1			
H5	− 0.18	0.09	0.163	0.222	0.165	1		
H6	− 0.35	− 0.1	− 0.31	− 0.36	− 0.31	− 0.18	1	
H7	0.05	0.145	− 0.14	0.06	− 0.16	0.247	0.05	1

solution is a much better indicator of task difficulty since hidden units index the number of dimensions or ‘cuts’ demanded by the problem space in order to solve the problem (Dawson, 1998).

Comparing network 3’s algorithm against network 2’s algorithm revealed some differences. For example, unlike the role of hidden unit 4 in network 2, hidden unit 4 in network 3 detected rules exclusively. However, hidden unit 4 in network 3 helped the other hidden units detect desired responses at their

Table 11
Definite features for bands from hidden units 0, 1, and 2 of network 3

Hidden unit	Band label	Definite features	n[a]
0	A	I0 ≠ I2, I1 = I3, I1 ≠ I7, I1 ≠ I8, I3 ≠ I7, I3 ≠ I8, I7 = I8	384
0	B	I0 ≠ I2	1152
0	C	I0 ≠ I2	1536
1	A	I0 ≠ I2	3072
2	A	I0 ≠ I2, I1 = I3, I1 ≠ I13, I1 ≠ I14, I3 ≠ I13, I3 ≠ I14, I13 = I14	384
2	B	I0 ≠ I2	1152
2	C	I0 ≠ I2	1536

‘≠’ indicates a perfectly negative correlation between input units; ‘=’ indicates a perfectly positive correlation between input units.

[a] n, number of patterns falling in each band.

respective ‘specific’ card locations. Therefore, network 3 provided further evidence of a ‘specialized’ algorithm that focused on card location. To be sure, network 3’s algorithm was slightly less complex if we look at the definite features detected by each of its hidden units. For example, the jittered density plots of network 3’s hidden units (with the exception of hidden units 3 and 7) were characterized by few bands. Only the jittered density plots of hidden units 3 and 7, as evidenced by their many bands, revealed a more intricate distinction among input patterns. Although network 3’s algorithm appeared less intricate as revealed by the banding, this evidence does not suggest that generating the ‘p’ and ‘q’ solution is more easily accomplished than generating the ‘p’ and ‘not-q’ solution. Number of bands is not a reliable indicator of task difficulty since it only reflects the pattern features falling into each of the decision regions or cuts made by the hidden units. Although the bands elucidate the definite features that the network used to generate a response, the bands do not by themselves reflect a task’s difficulty. The number of hidden units required by the network to solve the task is a better indicator of a task’s difficulty.

6. General discussion

Our purpose in this paper was to explore a connectionist account of performance on Wason’s selection task. We attempted to meet this goal by illustrating how three different PDP networks generated different solutions to the task. Two networks were trained to generate common but only partially correct responses (i.e., the ‘p’ card alone and both the ‘p and q’ cards) and one network was trained to generate the fully correct response (i.e., ‘p and not-q’).

Results from training these networks suggested that selecting the ‘p’ and ‘not-q’ response was more difficult than selecting only the ‘p’ response. The difficulty was reflected in the greater number of hidden units required by network 2 to learn to select the ‘p’ and ‘not-q’ in response to the task. It was possible, however, that network 2 required eight hidden units to learn the task not because the task was solved by selecting two cards but because selecting the ‘not-q’ response was inherently difficult, as has been found with human participants. We settled this confound by training a third network to select both the ‘p’ and ‘q’ cards in response to the task.

The results obtained from training network 3 revealed that selecting the ‘p’ and ‘q’ response was as difficult as selecting the ‘p’ and ‘not-q’ response. Network 3 also required eight hidden units to converge to a solution (as did network 2). Although the algorithm that network 3 generated with these eight hidden units was characterized by the detection of fewer input features (i.e., network 3’s jittered density plots of hidden unit activity revealed fewer bands, suggesting fewer of the pattern’s definite features needed to be discriminated for a solution) compared to network 2’s algorithm, banding results are not indicative of task difficulty.

Unlike number of hidden units, bands are not indicative of a task’s difficulty; the greater the number of bands one observes in a network’s

Table 12
Definite features for bands from hidden units 3 and 7 of network 3

H3-bands	Definite features	n[a]	H7-bands	Definite features	n
A	I0 ≠ I2, I1 = I3, I1 ≠ I4, I1 ≠ I5, I3 ≠ I4, I3 ≠ I5, I4 = I5	384	A	I0 ≠ I2	1632
			AB	I0 = 0, I1 = 1, I2 = 1, I3 = 0, I4 = 0, I5 = 1	96
B	I0 ≠ I2	1056			
C	I0 = 1, I1 = 1, I2 = 0, I3 = 0, I4 = 0, I5 = 0	96	AC	I0 = 0, I1 = 1, I2 = 1, I3 = 0, I4 = 0, I5 = 0	96
D	I0 = 1, I1 = 1, I2 = 0, I3 = 0, I4 = 0, I5 = 1	96	AD	I0 = 1, I1 = 1, I2 = 0, I3 = 0, I4 = 0, I5 = 0	96
E	I0 ≠ I2, I0 = I4, I1 = I5, I2 ≠ I4	288	AE	I0 = 1, I1 = 1, I2 = 0, I3 = 1, I4 = 0, I5 = 1	96
F	I0 ≠ I2, I0 ≠ I4, I0 = I5, I1 = 0, I2 = I4, I2 ≠ I5, I3 = 1, I4 ≠ I5	192	B	I0 = 1, I1 = 1, I2 = 0, I3 = 1, I4 = 0, I5 = 0	96
G	I0 = 0, I1 = I3, I1 = I5, I2 = 1, I3 = I5, I4 = 1	192			
			C	I0 ≠ I1, I0 ≠ I2, I0 ≠ I3, I0 = I5, I1 = I2, I1 = I3, I1 ≠ I5, I2 = I3, I2 ≠ I5, I3 ≠ I5, I4 = 0	192
H	I0 ≠ I2, I0 = I4, I2 ≠ I4	384			
I	I0 = 0, I1 = 1, I2 = 1, I3 = 0, I4 = 1, I5 = 0	96	D	I0 = 1, I1 = 0, I2 = 0, I3 = 0, I4 = 0, I5 = 0	96
J	I0 = 1, I1 = I3, I1 = I5, I2 = 0, I3 = I5, I4 = 0	192	E	I0 = 0, I1 = 0, I2 = 1, I3 = 0, I4 = 0, I5 = 0	96
K	I0 = 1, I1 = 1, I2 = 0, I3 = 0, I4 = 1, I5 = 0	96	F	I0 = 1, I1 = 0, I2 = 0, I3 = 0, I4 = 0, I5 = 0	96
			G	I0 = 1, I1 = 0, I2 = 0, I3 = 1, I4 = 0, I5 = 0	96
			H	I0 = 1, I1 = 1, I2 = 0, I3 = 1, I4 = 0, I5 = 0	96
			I	I0 = 1, I1 = 1, I2 = 0, I3 = 0, I4 = 0, I5 = 1	96
			J	I0 = 1, I1 = 1, I2 = 0, I3 = 0, I4 = 1, I5 = 0	96
			K	I0 = 0, I1 = 1, I2 = 1, I3 = 0, I4 = 0, I5 = 0	96

‘≠’ indicates a perfectly negative correlation between input units; ‘=’ indicates a perfectly positive correlation between input units or it can indicate that an input unit takes on a specific value or definite feature (e.g., I0 = 1).

[a] n, number of patterns falling in each band.

solution does not signal a greater complexity of the task being solved. The reason for this can be illustrated with the following scenarios:

Person A has little experience in computer programming but needs to create a computer program that will keep track of household expenses.

Table 13
Definite features for bands from hidden units 4, 5, and 6 of network 3

Hidden unit	Band label	Definite features	n[a]
4	A	I0 = I1, I0 ≠ I2, I0 ≠ I3, I1 ≠ I2, I1 ≠ I3, I2 = I3	768
4	B	I0 ≠ I1, I0 ≠ I2, I0 = I3, I1 = I2, I1 ≠ I3, I2 ≠ I3	768
4	C	I0 ≠ I2, I1 = I3	1536
5	A	I0 ≠ I2	2688
5	B	I0 ≠ I2, I1 = 0, I5 = 0	288
5	C	I0 = 1, I1 = 0, I2 = 0, I3 = 0, I4 = 0, I5 = 0	96
6	A	I0 ≠ I2, I1 = I3, I1 ≠ I10, I1 ≠ I11, I3 ≠ I10, I3 ≠ I11, I10 = I11	384
6	B	I0 ≠ I2	1152
6	C	I0 ≠ I2	1536

'≠' indicates a perfectly negative correlation between input units; '=' indicates a perfectly positive correlation between input units or it can indicate that an input unit takes on a specific value or definite feature (e.g., I0 = 1).

[a] n, number of patterns falling in each band.

> *The algorithm he or she uses to create the computer program is lengthy and intricate where every step is included and its execution detailed* (*as remembered from a course in programming that he or she took years ago*).

Now consider:

> *Person B has extensive experience in computer programming and needs to create a computer program that keep track of household expenses. The algorithm he or she uses will not likely be as intricate as the algorithm used by person A. The reason for this is that person B's knowledge in programming will simplify the algorithm to include only the most fundamental and necessary features, with the details at each step being already automated in procedural memory.*

Given both these scenarios, is it possible to evaluate how difficult it is to create a computer program that will keep track of household expenses? Not really because the algorithms developed by person A and B are more indicative of each person's knowledge in the task domain than of the structural difficulties of creating a program to keep track of household expenses. Analyzing 'banding' in a value unit architecture provides clues as to the algorithm developed by the network to solve the task. These clues are important in order to understand how the task is solved by the network. However, banding does not provide an index of task difficulty. The algorithm used to solve a task and the task's difficulty are not necessarily the same thing.

A possible criticism of the present approach is that the results we obtained are likely to be very dependent on the representation of the task. This charge, however, can be directed at most selection task studies that manipulate contextual variables — the responses are dependent on the specific contextual representation of the task. How investigators present (or represent) a problem to human participants (or networks) will undoubtedly influence the responses made. The problem of representation has in large part been the source of experimental manipulations with the selection task since how the task is presented to participants has been shown to alter their behavior (and the inferences made about reasoning competence). For example, presenting the task within a rich contextual framework leads to better 'logical' performance than when the task is presented within an impoverished contextual framework.

It is because of the multitude of algorithmic theories about selection task performance that we attempted to explore an architectural account. Collectively, the results obtained from training networks 1, 2, and 3 provide a different perspective of performance on the selection task. First, we found that all three networks solved the task by focusing on card location. This focus on card location suggests that the cards or the evidence from which to test the rule in the selection task may be important to explain participants' performance. Few theories have focused on how participants specifically encode and interpret the nature of the evidence in the selection task. To be sure, pragmatic reasoning theory, social contract theory, and mental models theory are all theories that can accommodate this specific dimension since they emphasize the interaction between the reasoner and the task. Nonetheless, this dimension has not yet been fully explored. We believe that future human studies of performance on the selection task should focus specifically on how participants view or encode the evidence with which they will test the rule.

Second, our results suggest that solving the selection task by selecting cards 'p' and 'q' is as difficult as selecting cards 'p' and 'not-q.' This is a controversial conclusion since the current literature leads us to believe that selecting the 'not-q' card is inherently difficult for human participants to initiate. For example, rule theorists suggest that people might not have the schemas to generate the 'not-q' response (Braine, 1978; Rips, 1994), whereas other theorists suggest that the 'not-q' response is obscured unless the task is framed in a meaningful context (for a review, see Evans et al., 1993). We think that the difficulty might be illusory, however; in principle generating the 'p' and 'not-q' cards might be just as easy (or difficult) as generating the 'p' and 'q' cards. The mediating variable determining how easy (or hard) it is to generate the 'p' and 'not-q' might be the 'correspondence' between the participant and the task; that is, the level of expertise or knowledge that the participant brings to the task.

One distinguishing feature between novices and experts is in the algorithms they develop to solve problems (Anderson, 1983). Experts tend to abstract simplifying features and generate more elegant solutions to problems, in comparison to novices. For example, novices develop what could be termed 'complex' algorithms in comparison to experts if only because novices fail to appreciate some of the simplifying conditions or 'general patterns' that experts readily see in the task (Anderson, 1983; de Groot, 1965, 1966; Jeffries, Polson, Razran & Atwood, 1977). Hence, depending on the sample of respondents tested, some solutions might appear to be generated easily and some solutions to be generated with difficulty — if at all. The 'not-q' response might appear more difficult for participants to select in abstract (i.e., lacking a meaningful context) versions of the selection task because respondents with little logical expertise are normally tested. A lack of logical expertise defines respondents in most studies of the selection task because investigators want to study 'natural, everyday' reasoning. In contrast, the 'not-q' response might be more easily generated in meaningful versions of the selection task because these versions normally use everyday contexts that allow respondents to use their everyday expertise and appreciate the appropriateness of the 'not-q' response. In short, we think that future reasoning studies need to incorporate knowledge or expertise as a variable in models of performance.

In conclusion, architectural accounts are an important contribution to inquiry because they not only provide fundamental support or counters to theoretical ideas but they are also a source of new theoretical ideas. According to Shastri (1991),

> [*C*]*onnectionism should not be viewed merely as an implementation paradigm. For adopting the connectionist paradigm forces us to revise many of our views about representation and reasoning. The most significant revisions stem from the fact that a connectionist system must operate without an interpreter*. (*p*. 282)

We hope to see future studies focus on how reasoners encode the selection task and, in particular, how they encode or interpret the evidence with which to test the rule. More generally, we hope to see more architectural accounts of reasoning performance in the literature so that the tools used to increase our understanding of cognitive phenomena do not remain stagnant.

Acknowledgements

The present research was funded by grants from the Social Sciences and Humanities Research Council of Canada (SSHRC) and the Natural Sciences and Engineering Research Council of Canada (NSERC).

The authors would like to thank Dr. Robert J. Sternberg for his comments on an earlier version of the paper.

References

Anderson, J. R. (1983). *The architecture of cognition*, Harvard University Press, Cambridge, MA.

Bechtel, W., & Abrahamsen, A. (1991). *Connectionism and the mind*, Blackwell, Cambridge, MA.

Berkeley, I. S. N., Dawson, M. R. W., Medler, D. A., Schopflocher, D. P., & Hornsby, L. (1995). Density plots of hidden value unit activations reveal interpretable bands. *Connection Science 7*, 167–186.

Braine, M. D. S. (1978). On the relation between the natural logic of reasoning and standard logic. *Psychological Review 85*(1), 1–21.

Byrne, R. M. J. (1989). Suppressing valid inferences with conditionals. *Cognition 31*, 61–83.

Cheng, P. W., & Holyoak, K. J. (1985). Pragmatic reasoning schemas. *Cognitive Psychology 17*, 391–416.

Cheng, P. W., & Holyoak, K. J. (1989). On the natural selection of reasoning theories. *Cognition 33*, 285–313.

Cosmides, L. (1989). The logic of social exchange: Has natural selection shaped how humans reason? Studies with the Wason selection task. *Cognition 31*, 187–276.

Dawson, M. R. W. (1998). *Understanding cognitive science*, Blackwell, Malden, MA.

Dawson, M. R. W., Medler, D. A., & 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., & Schopflocher, D. P. (1992). Modifying the generalized delta rule to brain networks of non-monotonic processors for pattern classification. *Connection Science 4*, 19–31.

de Groot, A. D. (1965). *Thought and choice in chess*, Mouton, The Hague.

de Groot, A. D. (1966). Perception and memory versus thought. In: Kleinmuntz, B. (Ed.), *Problem-solving*, Wiley, New York.

Derthick, M. (1991). Finding a maximally plausible model of an inconsistent theory. In: Barnden, John A., & Pollack, Jordan B. (Eds.), *High-level connectionist models. Advances in connectionist and neural computation theory*, vol. 1, Ablex, Norwood, NJ, pp. 241–248.

Evans St., B. T. J., Newstead, S. E., & Byrne, R. M. (1993). *Human reasoning: the psychology of deduction*, Lawrence Erlbaum, Hillsdale.

Garnham, A., & Oakhill, J. (1994). In: *Thinking and reasoning*, Blackwell, Cambridge, MA.

Gobet, F., & Simon, H. A. (1998). Pattern recognition makes search possible: comments on Holding (1992). *Psychological Research 61*, 204–208.

Goldstone, R. L., & Barsalou, L. (1998). Reuniting perception and conception. *Cognition 65*, 231–262.

Jeffries, R. P., Polson, P. G., Razran, L., & Atwood, M. (1977). A process model for missionaries-cannibals and other river crossing problems. *Cognitive Psychology 9*, 412–440.

Johnson-Laird, P. N. (1983). Inference and mental models. In: *Mental models. Towards a cognitive science of language, inference, and consciousness*, Harvard University Press, Cambridge, MA.

Johnson-Laird, P. N., & Byrne, R. (1991). *Deduction*, Lawrence Erlbaum, Hillsdale, NJ.

Klayman, J., & Ha, Y. -W. (1987). Confirmation, disconfirmation, and information in hypothesis testing. *Psychological Review 94*, 211–228.

Liberman, N., & Klar, Y. (1996). Hypothesis testing in Wason's selection task: social exchange, cheating detection, or task understanding. *Cognition 58*, 127–156.

Margolis, H. (1987). *Patterns, thinking, and cognition*, The University of Chicago Press, Chicago, IL.

Moorehead, I. R., Haig, N. D., & Clement, R. A. (1989). An investigation of trained neural networks from a neurophysiological perspective. *Perception 18*, 793–803.

Oaksford, M., & Chater, N. (1993). Reasoning theories and bounded rationality. In: Manktelow, K. I., & Over, D. E. (Eds.), *Rationality: psychological and philosophical perspectives*, Routledge, London.

Rips, L. J. (1994). In: *The psychology of proof*, The MIT Press, Cambridge, MA.

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

Shastri, L. (1991). The relevance of connectionism to AI: a representation and reasoning perspective. In: Barnden, John A., & Pollack, Jordan B. (Eds.), *High-level connectionist models. Advances in connectionist and neural computation theory*, vol. 1, Ablex, Norwood, NJ, pp. 259–283.

Wason, P. C. (1966). Reasoning. In: Foss, B. M. (Ed.), *New horizons in psychology*, Penguin, London.