

# Learning Mechanisms which Construct Neighbourhood Representations

*Chris Thornton*  
Cognitive and Computing Sciences  
University of Sussex  
Brighton  
BN1 9QH  
UK

Email: Christopher.Thornton@firenet.uk.com  
Tel: (44)1273 678856

May 21, 2003

## **Abstract**

Learning is currently the focus of much research activity in cognitive science. But, typically, this research is oriented towards either the symbol-processing paradigm or the connectionist paradigm and therefore tends to generate models of two quite different types. A satisfactory *theory* of learning will, presumably, deal with the phenomenon in general terms rather than in terms of two special cases, so there would appear to be a need to try to identify abstractions which generalise the two types of model typically produced. The aim of the present paper is to identify one such abstraction. It puts forward a framework in which the behaviours of two symbol-processing learning mechanisms are directly commensurable with the behaviours of three connectionist learning mechanisms and shows that there is at least one theoretical constraint affecting *all* mechanisms covered by the framework.

## **1 Introduction**

At the present time computational research on learning falls into two broad categories. Firstly, there is research concentrating on the learning properties of symbol processing (SP) mechanisms, e.g. [1,2]; secondly there is research concentrating on the learning properties of parallel distributed processing (PDP) mechanisms, e.g. [3]. The first category of research is in the traditional AI paradigm; the second, the connectionist paradigm.

At the implementation level, connectionist learning mechanisms have little in common with symbol-processing learning mechanisms. In order to perceive any commonality at all it is necessary to move to a more abstract view of the learning task where implementation details are effectively hidden. One framework providing such a view has been presented by Valiant [4,5]. Another has been presented by Michalski [6]. In Michalski's framework, learning is defined as the process of **constructing or modifying representations of what is being experienced**. [6, p. 10]. This seems to apply quite well to both PDP and symbol processing systems, c.f., [3,7,8]; but although the definition is very general it is also, necessarily, quite vague. It has quite different meanings depending on the interpretation we give to **representation**.

In the present paper, we explore the consequences of assuming that **representation** as used in the Michalski definition means something specific, namely **neighbourhood representation**. We first describe what sort of structure a neighbourhood representation is. We then try to show that the behaviour of two symbol processing mechanisms and three PDP mechanisms can be interpreted in terms of its construction. Finally, we argue that this particular strategy is subject to a constraint, and that the mechanisms which exploit it are therefore limited in terms of what particular types of experience (i.e. input) they can represent.

The remainder of the paper is organised as follows. In the second section, the neighbourhood representation scheme is characterised and defined. Section three presents an illustrative example of the scheme. Sections four and five argue (respectively) that the behaviour of both the Classification algorithm [7] and the Candidate-Elimination algorithm [9] can be understood in terms of the construction of neighbourhood representations. The following three sections put forward similar arguments for the Perceptron [10], the Pattern Associator [11] and the Back-Propagation mechanism [3]. Section nine argues that mechanisms which can only form neighbourhood representations are unable to effectively represent certain classes of input. The final section presents a summary of the points made.

## 2 Neighbourhood representations

The word **representation** has been used in AI to denote computational structures of various different types. Specific examples can be characterised in terms of the type of relationship which is presumed to hold between the representation in question, and the represented object or objects. Let us assume that  $C$  denotes an arbitrary set of objects and that  $c$  denotes a representation of  $C$ . Now,  $c$  may represent  $C$  merely by virtue of the fact that we imbue it with a certain interpretation; c.f., a list of names considered as a representation for a group of people. Alternatively,  $c$  may represent  $C$  in a more formal way: it may be a program which generates  $C$ , or it may be a predicate which determines whether some arbitrary object is a member of  $C$ .<sup>1</sup>

---

<sup>1</sup>In this latter case,  $c$  forms a *concept* for the class.

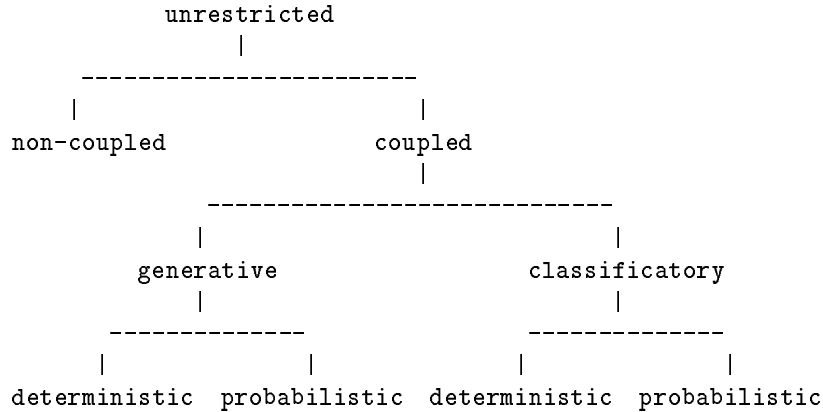


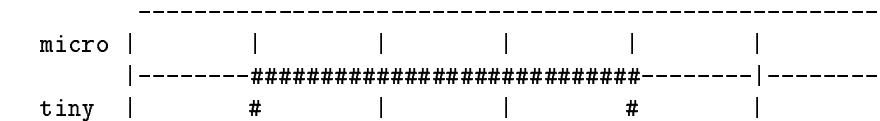
Figure 1:

Let us say that *c* is **coupled** only if it has a formal, computational relationship with *C*. Let us say that it is **classificatory** if it forms a predicate which correctly classifies descriptions according to whether or not they are in *C*, and that it is **generative** if it is a program which generates *C*. Finally, let us say that it is **deterministic** if it generates/classifies (elements of) *C* perfectly, or **probabilistic** if it does so to some given level of accuracy. These distinctions are summarised in the taxonomy shown in Figure 1.

We now define a *neighbourhood representation* as a coupled representation of a certain type. *D* will be assumed to denote the data language.<sup>2</sup> *C* will continue to label the set of elements which is represented by the hypothetical representation *c*. We will say that any coupled representation forms a neighbourhood representation if there is an interpretation of *c* in which the represented set *C* is defined in terms of the points inside *m* non-degenerate regions of an *n*-dimensional space.

### 3 Example

Imagine that a data element is a 2-tuple with the first component drawn from the set {huge vast big tiny med micro} and the second component drawn from the set {moped bike jet car prop glider}. Possible elements are then pairs such as <big car> and <big bike>.



<sup>2</sup>In concept learning work *D* would normally be called the description language. In connectionist work, it would be called the input language or input space.

	-----#----- ----- -----#----- -----
med	#                                #
	-----#----- ----- -----#----- -----
big	#                                #
	-----##### ----- -----##### -----
huge	
	----- ----- ----- ----- -----
vast	
	----- ----- ----- ----- -----
	bike      moped      car      prop      jet      glider

C is a set of nine 2-tuples:

```
{<big moped> <med car> <med moped>
 <tiny car> <big prop> <med prop>
 <tiny moped> <big car> <tiny prop>}
```

Consider the case where each set of terms forms the sequence of dimensional values for one dimension of the 2-dimensional space shown in Figure 1. In this context, C can be represented in terms of the points inside a single region of the space. The region might be defined in terms of the 2-tuple corresponding to its top, left cell, and the 2-tuple corresponding to its bottom-right cell; i.e. <tiny moped> and <big prop>. A representation of this type counts as a neighbourhood representation because it defines C in terms of the points inside m regions of the space (with m = 1).

## 4 Decision trees as neighbourhood representations

We now turn attention to the task of showing that the behaviour of some learning mechanisms can be interpreted in terms of the *construction* of neighbourhood representations. We will first consider two symbol processing mechanisms. Later we will consider some PDP mechanisms.

One of the largest families of symbol processing learning mechanisms is derived from Quinlan's ID3 program [12, Quinlan, Learning from 13, 7, 14, 15]. The main component in this program is the Classification algorithm [16] which has roots in the Concept Learning System of Hunt, Marin and Stone [17]. The Classification algorithm builds a decision tree characterising a given class of elements (i.e. instances). The tree can be interpreted as a classification rule or as a concept definition. The data (description) language for the Classification algorithm is a feature space and dimensions of this space range over sets of mutually exclusive feature values. If a data element has n attributes then there are n mutually exclusive feature sets and n dimensions in the feature space.

Consider the case where there are just two mutually exclusive sets of attributes: {red green blue} and {large small medium}. These sets contain the

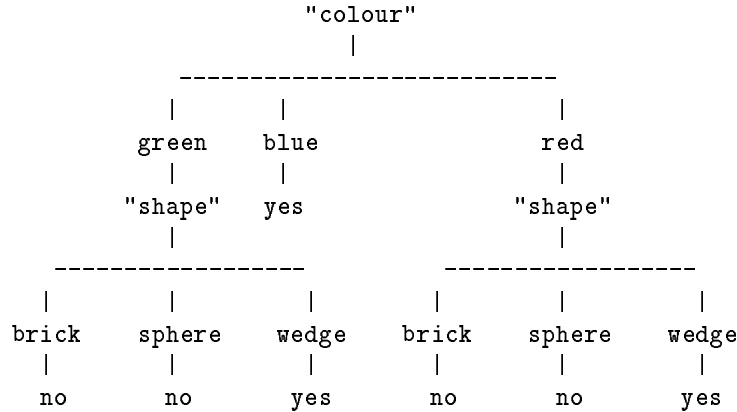


Figure 2:

possible values for a `colour` feature and a `size` feature respectively. The set  $C$  which we want to represent is known to contain the pairs  $\langle \text{blue wedge} \rangle$ ,  $\langle \text{green wedge} \rangle$ ,  $\langle \text{blue brick} \rangle$ ,  $\langle \text{red wedge} \rangle$ , and  $\langle \text{blue sphere} \rangle$ ; but to exclude  $\langle \text{red brick} \rangle$  and  $\langle \text{green sphere} \rangle$ . The sort of decision tree which might be learned by the Classification algorithm presented with the set of included and excluded elements (i.e. positive and negative examples) is shown in Figure 2. It corresponds to the disjunctive rule

`blue or (green and wedge) or (red and wedge)`

which might be abbreviated to

**blue or wedge**

For this to count as a neighbourhood representation it must be the case that there is an interpretation of the rule in which  $C$  is defined in terms of the points inside  $m$  regions of some  $n$ -dimensional space. Note that the exclusion of a specific feature value from a conjunct in a rule of the type shown captures the fact that elements of  $C$  may have *any* value for that feature. Thus each conjunct effectively identifies a certain region of the feature space. The boundaries of the region in any dimension (feature) which is unspecified enclose all the values in the dimension. The boundaries in any dimension which is specified enclose just the specified value.<sup>3</sup> Thus the rule `(blue or wedge)` which correctly classifies elements of  $C$ , can be interpreted as representing  $C$  in terms of a set of points within two rectangular regions of the feature space as shown in Figure 3. Descriptions which are known to be in  $C$  are shown as plus symbols. Descriptions which are known to be excluded are shown as minus symbols.

---

<sup>3</sup>The sequencing of dimensional values in this case is irrelevant.



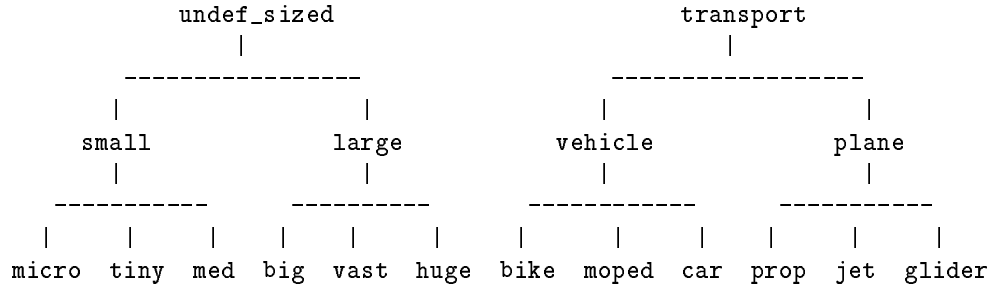


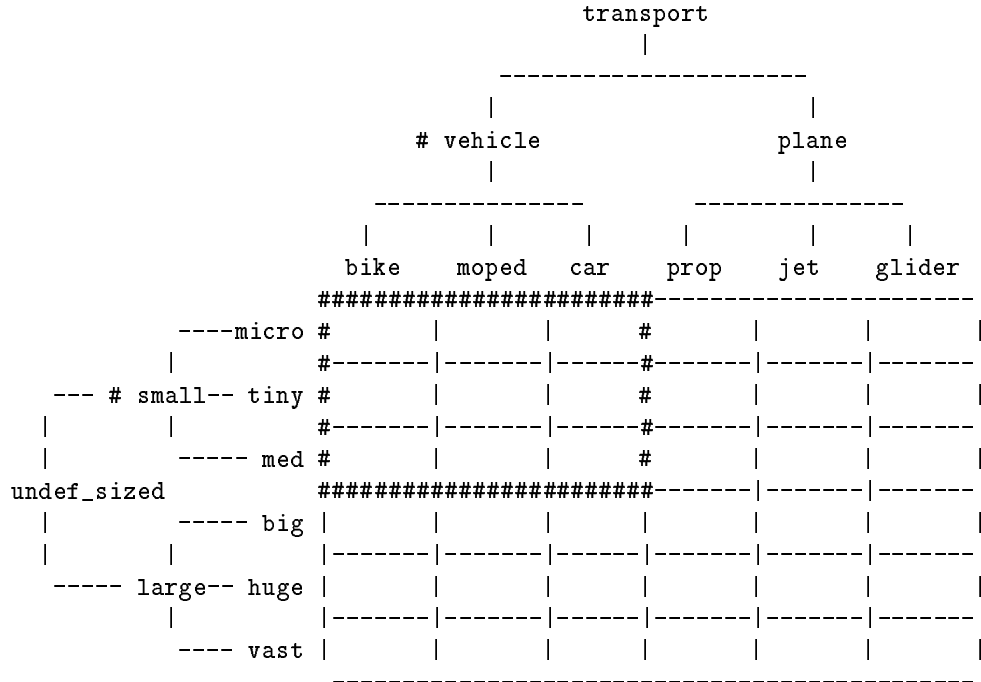
Figure 4:

jet>. Representations are 2-tuples which may contain non-terminals; e.g. <large bike>. Note that in an n-dimensional space where each boundary ranges over the left-to-right sequence of the leaf nodes from one of the n generalisation hierarchies, specific elements correspond to specific cells of the space. Moreover, each term in a representation effectively defines a subrange of the corresponding dimension. In this context, a complete representation defines a rectangular (in general, a hyper-rectangular) region of the space. This region encloses all the cells corresponding to covered elements; see Figure 5. We can, therefore, legitimately interpret a representation constructed by the Candidate-Elimination algorithm (in the case where the instance language is defined using n generalisation hierarchies) as a neighbourhood representation [24].

## 6 The Perceptron

Attention now turns from symbol processing learning mechanisms to PDP mechanisms. For present purposes, a **PDP learning mechanism** will be taken to be a combination of some specific network architecture and some specific learning rule. In fact, we will only consider mechanisms which use the delta learning rule (in either its generalised or ungeneralised form) [3, chapters 2 and 8]. We will look at both the Pattern Associator (also known as the associative net) and the back-propagated feed-forward net which we will refer to as the Back-Propagation mechanism. But first we will consider a very simple case: that of the Perceptron [10]. This mechanism has been subjected to a rather rigorous analysis by Minsky and Papert [25] and in fact the interpretation of its behaviour which we will develop is close to the usual one for this mechanism; c.f., [26].

The essential architecture of the Perceptron is very simple. A number of input units are connected to one output unit and each connection is associated with a weight. The set of weights is referred to as the weight vector. Elements of the data language (i.e. inputs) are sets of activation values for the input units (i.e. input vectors) and each activation value is just a real number between 0 and 1. The output unit is associated with a threshold and produces a 1 if the



Region enclosing data elements (i.e. cells) covered by <small vehicle>.

Figure 5:

inner product of the input vector and the weight vector (i.e. the sum produced by multiplying each input value with the appropriate connection weight) exceeds the threshold; otherwise it outputs a 0. The Perceptron has derived a classificatory representation for some set C if it always outputs a 1 whenever the input vector is in C, and 0 in all other cases. This representation consists of the weight vector and the threshold.

Since the elements of D are n-tuples of real numbers they are very easily interpreted as points in an n-dimensional space. In this case the weight vector and threshold form a hyperplane separating the inputs which produce a 1 output from the inputs which produce a 0 output [25]. This representation effectively treats the data (input) language as a space and defines the represented class C in terms of the points in a single region of this space; i.e. the region falling to one side of the hyperplane. Representations constructed by the Perceptron are therefore legitimately viewed as neighbourhood representations.

## 7 The Pattern Associator

The Pattern-Associator (or associative net) is closely related to the Perceptron; its basic architecture is quite similar, but whereas, in the Perceptron, the  $n$  input units are connected to a single output unit, in the Pattern Associator, the  $n$  input units are connected to a *set* of  $m$  output units. Furthermore, there is no thresholding. The output of the mechanism for a given input is the set of activations of the output units.

In general, the Pattern-Associator is construed as a mechanism which forms associations between input and output patterns (hence the name). However, for present purposes we are interested in viewing its behaviour in terms of the construction of coupled representations. We will say that a Pattern-Associator has formed a classificatory representation for some set of inputs  $C$  if it always gives the same output when presented with an element of  $C$ , but gives a different output in all other cases. The activation of a given output unit for a given input is the inner product of the input vector with the output unit's weight vector. If vectors are normalised the inner product of two vectors is just the projection of one on the other [3]. This means that we can think about the relationship between an input vector and a given activation of an output unit in geometric terms.

missing file pa\_1invec.di

missing file pa\_2invec.di

missing file pa\_3invec.di

Let us consider the special case where the Pattern-Associator has only one output unit  $u$  and input vectors have only two components. We know that the activation of  $u$  for some particular input is just the projection of the input vector on  $u$ 's weight vector. This means that, to achieve, some particular activation ( $O_d$  for a given input,  $u$ 's weight vector must be positioned such that its tip lies somewhere along a perpendicular bisecting the input vector at a point corresponding to the desired output; see figure above. (Note that the vectors in the diagrams are not drawn to scale.) To achieve some particular output for two different input vectors, the tip of the weight vector must lie along two different perpendiculars (i.e. the perpendiculars which strike the input vectors at the point corresponding to ( $O_d$ )). This means that the tip of the weight vector must be positioned at the intersection of the two perpendiculars; see figure above.

missing file pa\_error1

In the case where we want  $u$  to give the same output for three different input vectors, there will be three perpendiculars and no unique point of intersection. However, the outputs provided by  $u$  will be as near as possible to ( $O_d$  if the total distance between the tip of the weight vector and the perpendiculars is minimised; see figure above. Application of the delta learning rule is, in fact,

guaranteed to achieve this effect [3]. The general implication is that the Pattern-Associator cannot form deterministic representations for more than two data elements (input vectors) in the case where elements have only two components. In general, it cannot form deterministic representations for more than  $n$  elements in an  $n$ -dimensional input space.<sup>5</sup>

Probabilistic representations are a possibility however. We can view the Pattern-Associator as having formed a representation for some set of elements  $C$  if the output given by  $u$  in the case where the input is in  $C$  is always within some small range of values. This allows us to view the weight vector shown in Figure 5 as a representation for a class  $C$  consisting of the three elements shown. The output produced for any input in  $C$  will always be within some given range centred on the desired value ( $O_d$ , as shown in Figure 5).

missing file pa\_gen1.di

The interesting thing about this representation is that it automatically generalises all vectors which lie between the given three inputs. The output by  $u$  will always be within the given range whenever there is a perpendicular through the input vector which (1) passes through the tip of the weight vector and (2) strikes the input vector at a point which is within the defined output range; see Figure 5. The set of elements satisfying this description form a region in the vector space; see Figure 5. Thus, in this context, we can interpret the representations formed by the Pattern-Associator as neighbourhood representations.

missing file pa\_gen2

## 8 The Back-Propagation Mechanism

We now turn attention to the back-propagated, feed-forward net [3, chapter 8], which we will refer to as the **Back-Propagation mechanism**. The architecture of this mechanism consists of a layer of input units which feed activation to  $n$  layers of hidden unit. Units in the last layer of hidden units feed activation to  $n$  output units. The Perceptron and Pattern-Associator correspond to special cases of the Back-Propagation mechanism. The Perceptron corresponds to the case where there is one output unit, no hidden units, and the activation function used is effectively a threshold (step) function. The Pattern-Associator corresponds to the case where there are  $n$  output units, no hidden units and activations are computed using a linear function. Since the Back-Propagation mechanism generalises both the Perceptron and the Pattern-Associator we can infer that it is capable of forming neighbourhood representations in data space in both Perceptron-like fashion and Pattern-Associator-like fashion. It turns out that these are not the only possibilities.

Let us consider the classic case in which Back-Propagation is used to construct a network which can compute the exclusive-or relation. This is, of course,

---

<sup>5</sup>This is just a manifestation of the fact that the Pattern-Associator can only learn to associate *linearly independent* patterns [3].

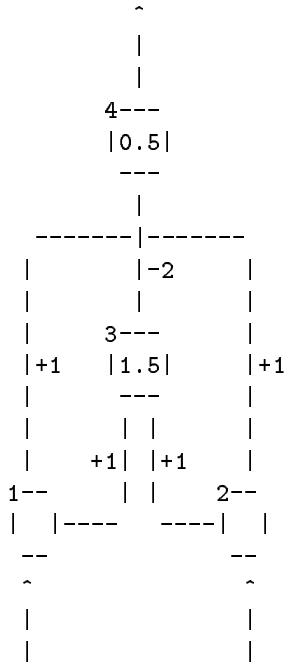
the paradigm example of a task which cannot be performed by the Perceptron [25]. We need not concern ourselves with the details of the learning mechanism (the generalised delta rule) since we are only interested in the form of the representations which are learned. In order to maintain uniformity, we will view a network which can compute exclusive-or as a network which has effectively constructed a representation for a certain subset of data elements (input vectors). That is to say, we will consider that the ability to compute the exclusive-or of pairs of binary inputs is the same as having a classificatory representation for the set  $\{<0,1>, <1,0>\}$  in the case where the data language is just the set of all pairs of binary digits.

Figure 6 is a representation of the sort of network which might be produced by Back-Propagation for the exclusive-or task (after [3, p. 321]). Small rectangles correspond to units: rectangles in the lowest layer are the input units, the rectangle in the middle layer is the (single) hidden unit while the unit in the top layer is the (single) output unit. Lines between rectangles correspond to links between units. The number appearing in the top, left corner of a rectangle is a label for the corresponding unit, while the number appearing within a rectangle is the unit's threshold. Numbers on lines are just the weights on the corresponding links.

The network works in a very simple way. The result of turning both input units on (i.e. of presenting the element  $<1,1>$ ), is that the hidden unit comes on and turns the output unit off. If an output of 0 is interpreted as implying that the input is *not* in the represented set, then the behaviour in this case corresponds to a correct classification decision. The result of turning just one input unit on and the other off is that the hidden unit remains off but the output unit comes on. The result of turning neither input unit on is that both the hidden unit and the output unit stay off. All of these cases correspond to correct classifications.

Is there any way of viewing the overall state of the network as a neighbourhood representation for the set  $\{<1,0>, <0,1>\}$ ? Each unit in the network can be in one of two states (either on or off). Thus we can view the set of possible states for the two non-input units in terms of a 2-dimensional space in which each boundary ranges over the possible states of one unit. If we then write into each cell in this space the data element which achieves the given state (of the two non-input units), we find that the elements in the represented set occupy contiguous cells. Obviously the representation implemented is made up of the connection weights. But in light of the remarks made above, we can see the weights as mapping states of the input units into the state-space for the non-input units in such a way that the input states corresponding to represented elements are mapped onto points in a *region* of the state-space.

Under this interpretation, an **on** state in the output unit is just an indication that the data element has instantiated a point in the state-space which is inside the defined region. Thus, there is an interpretation of the representation in which the elements in the represented set  $C$  are defined in terms of the points inside a region of the activation space for a subset of (non-input) units; see Figure 7. The overall conclusion is that, in this particular case, the represen-



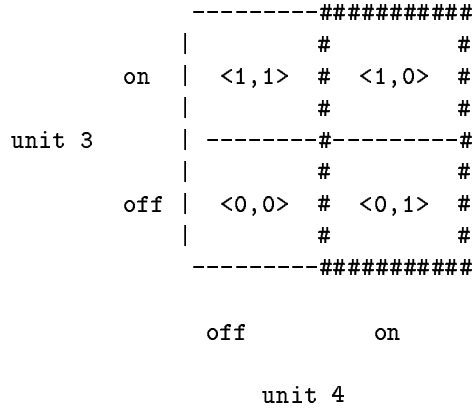
Exclusive-OR Network (after [3, p. 321])

Figure 6:

tation constructed by Back-Propagation forms a neighbourhood representation and that it does so in an activation space for a subset of non-input units.<sup>6</sup>

---

<sup>6</sup>Broomhead and Lowe [27] have shown recently that network architectures developed by Back-Propagation can be viewed as devices for the interpolation of data in multi-dimensional space (p. 346) using radial basis functions. The notion that the Back-Propagation mechanism can form neighbourhood representations in unit-activation space may turn out to be an informal characterisation of their result.



The state space for units 3 and 4. Numbers in cells represent the input pairs which achieve the given state.

Figure 7:

## 9 NETtalk

A compelling demonstration of the power of the Back-Propagation mechanism is presented in [28]. This describes how Back-Propagation was used to train a network to produce correct pronunciations (or at least descriptions of correct pronunciations) for words in a large corpus of text. The network used had one layer of hidden units. Inputs were effectively descriptions of the letters in a small, moving *window* onto the corpus of text. Outputs were descriptions of the correct pronunciation for the letter in the centre of the window.

The network achieved a rather impressive level of performance on this task. We can view its final state as a kind of complex, classificatory representation for a large set of classes. We see each of the  $n$  possible outputs as the name of a class of input (i.e. the class of inputs which all require a certain pronunciation), and we see the output in a specific case as classifying the input in terms of these classes.

Can these classificatory representations be viewed as neighbourhood representations? The answer is not straightforward. The number of hidden units in the network mean that the sort of approach taken in the case of the exclusive-or network is out of the question. However, some insight can be gained by considering the results of the hierarchical cluster analysis which the authors applied to the system. The aim of this analysis was to try to determine whether the network had constructed an internal representation of the mapping between specific letters and specific sounds. The analysis was carried out in terms of *functional vectors* each of which corresponded to a particular transformation implemented by the network [29].

Where did these functional vectors come from? Note that in any particular

case, the network accepts an input and produces an output. In the process, all the hidden units take on a given activation. Each activation is just a real number between 0 and 1 so the set of activations in the  $n$  hidden units can conveniently be viewed in terms of a point in an  $n$ -dimensional space. The states of the hidden part of the network produced by each of the possible inputs thus form a set of points in an  $n$ -dimensional space.

Now, in each particular learning experiment, the network arrived at a different global state. But this does not imply that in each case it arrived at a different representation for the associations between letters and correct pronunciations. The representation for a particular association is just the functional vector involved in the implementation of that association. And the representation for the set of associations is the complete set of functional vectors. Application of hierarchical cluster analysis to the functional vectors for the NETtalk networks showed quite clearly that, in each case, the learning procedure produced the *same* internal representation. Moreover it showed clearly that the representation was a **spatial** one: each letter/sound association was represented by a particular functional vector and the closeness of one vector to another symbolised to the degree of similarity between the corresponding associations [28].

In effect, the learning procedure produced a representation in which regions of functional-vector space symbolised particular classes of letter/sound association. A dramatic demonstration of this was the fact that in the representation, the vector space was partitioned into two distinct regions, with one region corresponding to the class of vowel pronunciations, the other corresponding to the class of consonant pronunciations [28]. Now, clearly, this sort of representation counts as a neighbourhood representation; but it is important to distinguish it from the type of representation constructed by the Perceptron and Pattern-Associator (data space representation) and by the Back-Propagation mechanism in the previous example (unit-activation space representation). The NETtalk system appears to construct neighbourhood representations in functional-vector space. Thus, overall, the Back-Propagation mechanism appears to be capable of constructing neighbourhood representations in at least three different ways. This implies that the theoretical constraints on Back-Propagation may not be as severe as they are in the case of the other mechanisms. It is to a consideration of one of these constraints that we now turn.

## 10 A theoretical constraint

The arguments presented above have tried to show that a variety of PDP and symbol processing learning mechanisms can be construed as forming neighbourhood representations. All but one of the mechanisms can be shown to do this *necessarily*. The single exception - the Back-Propagation mechanism - can be shown to construct neighbourhood representations in certain cases and to be capable of doing so in more than one space. Given the wide exploitation of this particular learning strategy, it is interesting to inquire about its generality and efficacy. In particular, we would like to know whether classes can always

be represented in terms of regions of points in some particular n-dimensional space. To begin with, however, we will consider a simpler question; namely, **can classes always be represented as subsets?** This generalises the original question since it is clearly the case that any class which can be represented in terms of m regions of points in some particular n-dimensional space can also be represented in terms of a particular subset of points.

Let us continue to use D to label the set of data elements and C to label the class of instances we are interested in. From here on, let us also use R to label the set of all representations and say that any class C is **concrete** with respect to D only if C is a subset of (?) D. Now, imagine a situation in which  $R = D$ , and D = the set of pairs which describe individual playing cards:

```
{ [3, hearts] [queen, diamonds] [9, spades] ... }
```

If C is the class of all black cards (i.e. the set of all pairs which describe black cards), then clearly, C is a subset of (?) D. C is classified as concrete with respect to D since it can certainly be represented in terms of a subset of D.

We now imagine the case where D remains the same but R is defined as the powerset of D. In this case classes may contain instances involving multiple elements of D. An example is the class of all **straights**; i.e. the set of all poker hands (5-tuples on D) which can be arranged as a continuous sequence of card values. We can define this class in terms of D but not in terms of any subset of D. We characterise this situation by saying that C is **abstract** with respect to D.

## 11 Very abstract classes

Now consider the class of all **close straights**. An instance of this class is a set of  $N$  hands such that they are all straights with fairly equal overall values. This class can be described in terms of playing cards but clearly not in terms of any subset of playing cards. Thus it is an abstract class with respect to  $D$ . But this is not the end of the story. Imagine  $D'$  is the set of all descriptions of complete poker hands (5-tuples on  $D$ ). Now, the class of all straights is abstract with respect to  $D$  but concrete with respect to  $D'$ . But the class of all close straights is abstract with respect to both. This suggests that the class of close straights is *more* abstract with respect to  $D$  than the class of all straights.

A formal model of this *abstractness* property is given in [30]. For present purposes, it is sufficient to note the implication that a class which can be defined in terms of some particular set of objects or descriptions may not necessarily be representable in terms of a *subset* of those objects. Since, as we have already noted, any neighbourhood representation represents a particular class in terms of a given subset we can infer that there will be cases in which a mechanism only capable of forming neighbourhood representations cannot learn representations for definable classes.

In addition to those definable classes which cannot be represented in terms of subsets, there are classes which *can* be represented in terms of subsets but not in terms of any set of non-degenerate regions. This may occur, for example, when a class is known to exclude certain negative instances as well as including certain positive instances. If, in the representation space, any region which would enclose more than one positive instance will inevitably enclose at least one negative instance then any neighbourhood representation will be degenerate. A situation fitting this description (in which the representation space is assumed to be identical to the data space) is depicted in Figure 8 (members and non-members of  $C$  are depicted as plusses and minuses in the usual way). In the concept learning literature, such situations are assumed to indicate the need for a disjunctive definition or, perhaps, an enhanced description language [8,23].

red			
green		-	+
blue		+	
	brick	sphere	wedge

Figure 8:

## 12 How the constraint applies to the Back-Propagation mechanism

We noted above that any mechanism which can only form neighbourhood representations will be incapable of forming a representation for any class which is abstract with respect to the representation language. We now consider the degree to which this constraint affects the Back-Propagation mechanism. There are various issues to be taken into account here. Firstly, as we have noted, Back-Propagation appears to be able to form neighbourhood representations in at least three different spaces: data (or input) space, unit-activation space, and functional-vector space. If  $S$  is the complete set of spaces in which Back-Propagation forms neighbourhood representations, then we might assume that its use of this learning strategy constrains it in the sense that it renders it unable to learn a representation for any class which is abstract (or undefinable) with respect to some member of  $S$ .

Unfortunately, this assumption is invalid since it is easily shown that Back-Propagation *can* form representations for classes which are abstract with respect to the data language. Note that each unit in the first layer of hidden units in a feed-forward network computes some function of the activations of the input units (i.e. some function of a data element). The activation of any given hidden unit will vary depending on what the input actually is and we can therefore regard it as implementing a probabilistic representation for a certain set of inputs; namely, the inputs which lead to the unit having a high activation.

The activation of a given unit in the second layer of hidden units depends on the activations of the units in the first layer. Thus, we can regard its activation as a representation for a subset of activation patterns on those units. Now, it will typically be impossible to represent the class of inputs represented by a given

unit in the *second* layer of hidden units in terms of any subset of D; thus the class represented is abstract with respect to D. The general conclusion, then, is that our original formulation of the constraint applying to Back-Propagation must be amended: if S is the complete set of spaces in which Back-Propagation forms neighbourhood representations, then its of this learning strategy constrains it in the sense that it renders it unable to learn a representation for any class which is abstract (or undefinable) with respect to any member of S except D.

### 13 Summary

The paper began by suggesting that the Michalski definition of learning provides a perspective from which the behaviour of symbol processing learning mechanisms is commensurable with the behaviour of PDP learning mechanisms. It provided some evidence in favour of this view by (1) making the assumption that the word **representation** as used in the definition can be interpreted as meaning **neighbourhood representation**, and by (2) describing two symbol processing mechanisms and three PDP mechanisms which can all be said to construct this sort of representation. Arguments have also been presented which show that any mechanism which can *only* construct neighbourhood representation will be, in principle, unable to represent certain classes of input (i.e. **experience**); and the degree to which this constraint applies to specific mechanisms was considered.

### References

- [1] Michalski, R., Carbonell, J. and Mitchell, T. (Eds.) (1983). *Machine Learning: An Artificial Intelligence Approach*. Palo Alto: Tioga.
- [2] Michalski, R., Carbonell, J. and Mitchell, T. (Eds.) (1986). *Machine Learning: An Artificial Intelligence Approach: Vol II*. Los Altos: Morgan Kaufmann.
- [3] Rumelhart, D. and McClelland, J. (1986). On learning the past tenses of english verbs. In D. Rumelhart, J. McClelland and the PDP Research Group (Eds.), *Parallel Distributed Processing: Explorations in the Microstructures of Cognition. Vols I and II*. Cambridge, Mass.: MIT Press.
- [4] Valiant, L. (1984). A theory of the learnable. *Communications of the ACM*, 27 (pp. 1134-42).
- [5] Valiant, L. (1985). Learning disjunctions of conjunctions. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence* (pp. 560-566). Los Altos: Morgan Kaufmann.
- [6] Michalski, R. (1986). Understanding the nature of learning: issues and research directions. In R. Michalski, J. Carbonell and T. Mitchell (Eds.),

*Machine Learning: An Artificial Intelligence Approach: Vol II* (pp. 3-24).  
Los Altos: Morgan Kaufmann.

- [7] Quinlan, J. (1983). Inferno: a cautious approach to uncertain inference. *The Computer Journal*, 26, No. 3.
- [8] Mitchell, T., Utgoff, P. and Banerji, R. (1983). Learning by experimentation: acquiring and modifying problem-solving heuristics. In R. Michalski, J. Carbonell and T. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach*. Palo Alto: Tioga.
- [9] Mitchell, T. (1977). Version spaces: a candidate elimination approach to rule learning. *Proceedings of the Fifth International Joint Conference on Artificial Intelligence* (pp. 305-310).
- [10] Rosenblatt, F. (1962). *Principles of Neurodynamics*. New York: Spartan Books.
- [11] McClelland, J. and Rumelhart, D. (1988). *Explorations in Parallel Distributed Processing: A handbook of Models, Programs, and Exercises*. Cambridge, Mass.: MIT Press.
- [12] Quinlan, J. (1986). Induction of decision trees. *Machine Learning*, 1 (pp. 81-106).
- [13] Quinlan, J. (1986). Learning from noisy data. In R. Michalski, J. Carbonell and T. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach: Vol II*. Los Altos: Morgan Kaufmann.
- [14] Quinlan, J. (1979). Discovering rules by induction from collections of examples. In D. Michie (Ed.), *Expert Systems in the Micro-Electronic Age* (pp. 168-201). Edinburgh: Edinburgh University Press.
- [15] Shapiro, A. and Niblett, T. (1982). Automatic induction of classification rules for a chess end-game. In M. Clarke (Ed.), *Advances in Computer Chess 3* (pp. 73-92). Oxford: Pergamon.
- [16] Bundy, A., Silver, B. and Plummer, D. (1985). An analytical comparison of some rule-learning programs. *Artificial Intelligence*, 27, No. 2 (pp. 137-81).
- [17] Hunt, E., Marin, J. and Stone, P. (1966). *Experiments in Induction*. New York: Academic Press.
- [18] Utgoff, P. and Mitchell, T. (1982). Acquisition of appropriate bias for inductive concept learning. *Proceedings of the Second National Conference on Artificial Intelligence* (pp. 414-417).
- [19] Winston, P. (1970). Learning structural descriptions from examples. AI-TR-231, Ph.D thesis, Cambridge, Mass.: MIT AI Lab.

- [20] Winston, P. (1975). Learning structural descriptions from examples. In P. Winston (Ed.), *The Psychology of Computer Vision*. McGraw-Hill.
- [21] Michalski, R. and Chilausky, R. (1980). Knowledge acquisition by encoding expert rules versus computer induction from examples: a case study involving soybean pathology. *Int. J. Man-Machine Studies*, 12 (pp. 63-87).
- [22] Dietterich, T., London, B., Clarkson, K. and Dromey, G. (1982). Learning and inductive inference. In P. Cohen and E. Feigenbaum (Eds.), *The Handbook of Artificial Intelligence: Vol III*. Los Altos: Kaufmann.
- [23] Utgoff, P. (1986). Shift of bias for inductive concept learning. In R. Michalski, J. Carbonell and T. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach: Vol II* (pp. 107-148). Los Altos: Morgan Kaufmann.
- [24] Thornton, C. (1987). Hypercuboid-formation behaviour of two learning algorithms. *Proceedings of the Tenth International Joint Conference on Artificial Intelligence* (pp. 301-303). Los Altos: Morgan Kaufmann.
- [25] Minsky, M. and Papert, S. (1988). *Perceptrons: An Introduction to Computational Geometry* (expanded edn). Cambridge, Mass.: MIT Press.
- [26] Forsyth, R. (1984). Machine learning strategies. In R. Forsyth (Ed.), *Expert Systems: Principles and Case Studies*. London: Chapman & Hall.
- [27] Broomhead, D. and Lowe, D. (1988). Multivariable functional interpolation and adaptive networks. *Complex Systems*, 2 (pp. 321-355).
- [28] Sejnowski, T. and Rosenberg, C. (1987). Parallel networks that learn to pronounce english text. *Complex Systems*, 1 (pp. 145-68).
- [29] Churchland, P. (1989). On the nature of theories: a neurocomputational perspective. In P. Churchland (Ed.), *The Neurocomputational Perspective*. Cambridge, Mass.: MIT Press.
- [30] Thornton, C. (1989). Defining higher-order classes. *Proceedings of the Second Scandinavian Conference on Artificial Intelligence*.