Representing part-whole hierarchies in connectionist networks

Geoffrey E. Hinton
Computer Science Department & Psychology Department, University of Toronto

Introduction

One reason for scepticism about connectionist models that use distributed representations is that there are currently no convincing demonstrations of how these models can represent complex, articulated structures. Drew McDermott (personal communication) has suggested that the approach would be far more convincing if it could come up with a sensible scheme for representing the meaning of a sentence such as: "She seems to be more at ease with her fellow students than with me, her adviser." This meaning is clearly composed of several major constituents with relationships between them, and each major constituent has its own, complex, internal structure. A representational scheme for dealing with meanings of this complexity must, at the very least, specify how the meanings of whole expressions are related to the meanings of their constituents and how it is possible, in some sense, to have the whole meaning in mind at once.

The example given above is typical of examples from many different domains. It appears that whenever people have to deal with complexity they impose part-whole hierarchies in which objects at one level are composed of inter-related objects at the next level down. In representing a visual scene or an everday plan or the structure of a sentence we use hierarchical structures of this kind. The main issue addressed in this paper is how to represent complex part-whole hierarchies in a connectionist network. Three different methods are described.

Symbols and the conventional implementation of hierarchical structures

It will be helpful to begin by reviewing the standard way of implementing hierarchical data-structures in a conventional digital computer. There are obviously many minor variations, but a suitable paradigm example is the kind of record structure that is found in languages like Pascal (but without the type constraints).

Each instance of a record is composed of a predetermined set of fields (sometimes called "slots" or "roles") each of which contains a pointer to the contents of the field which may be either another instance of a record, or a primitive object. Since the pointers can be arbitrary addresses, this is a very flexible way of implementing a hierarchical data-structure, but the flexibility is bought at the price of the von Neumann bottleneck: The addressing mechanism means that only one pointer can be followed at a time. [1]

The essence of a symbol is this: It is a small representation of an object that provides an "remote access" path to a fuller representation of the same object. [2] For example, the address of a record structure is a small representation and the whole record that it points to is a fuller representation. In general, this fuller representation is itself composed of small representations (e. g. addresses) of the structures that fill the fields of the record. Because a symbol is small, many symbols can be put together to create a "fully-articulated" representation of some larger structure and the size of this fully-articulated represention need not be any larger than the fully-articulated representations of its constituents.

When addresses are used as symbols, there is normally a purely arbitrary relationship between the internal structure of a symbol and the fully articulated representation that it provides access to. Looking at the individual bits in the symbol provides no information about what it represents. Occasionally this is not quite true. If, for example, one type of data-structure is kept in the top half of memory and another type in the bottom half, the first bit of a symbol reveals the type of the data-structure that it provides access to. So it is possible to check the type without following the pointer. This trick can obviously be extended so that many of the bits in

[1] Architectures such as the Connection Machine (Hillis, 1985) use routing hardware that allows many pointers to be followed at once, but even with hardware support, the routing is quite slow.

[2] There is, of course, much debate about the meaning of the word "symbol". The informal definition given here emerged from conversations with Allen Newell.

a symbol convey useful information. A symbol can then be viewed as a "reduced description" of the object.

The conclusion of this paper is that patterns of activity in parts of a connectionist network need to exhibit the double life that is characteristic of symbols. The patterns must allow remote access to fuller representations, but so long as the patterns are also reduced descriptions this remote access need only be used very occasionally (e.g. a few times per second). Most of the processing can be done by parallel constraint-satisfaction on the patterns themselves.

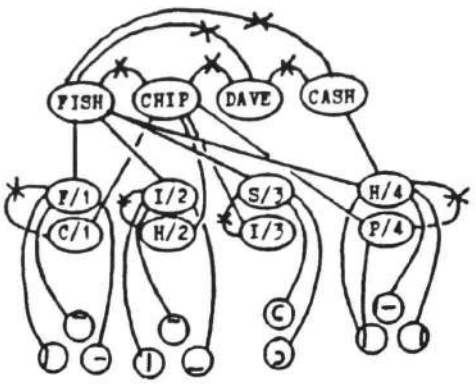

Figure 1: Part of a network used for recognizing words. Only a few of the units and connections are shown. The connections between alternative hypotheses at the same level are inhibitory.

Method 1: The fully-parallel implementation

Perhaps the most obvious way to implement part-whole hierarchies in a connectionist network is to use the connections themselves as pointers. Figure 1 shows an example taken from the work of Rumelhart and McClelland (1981). It is a network that recognizes a word when given partial information about the features of the letters in the word. We use this as the standard, concrete example of a part-whole hierarchy because it has clearly defined levels and the parts have convenient names, but this paper is not about word recognition. Because each relationship in the hierarchical tree-structure is implemented by its own dedicated connection, it is possible to do a lot of parallel processing during recognition. Simultaneously, many different letters can check whether the features they require are present, and many different words can check whether the letters they require are present.

One very important aspect of the Rumelhart and McClelland network is that each of the letter units has to be replicated for each of the four possible positions of a letter within the word (they restrict themselves to four-letter words). There is a separate unit for an H as first letter and an H as second letter. All the letter features and all the knowledge about which combinations of letter features make an H must also be replicated for each of the four positions. This replication is a natural consequence of implementing part-whole relationships with pairwise connections. A part-whole relationship involves three different things: The part, the whole, and the role that the part plays within the whole. In the conventional implementation using pointers, the role is encoded by which field the pointer is in. A pairwise connection between neuron-like units does not have anything equivalent to a field, and so the destination of the connection is used to represent both the field and the contents of the field. Thus, instead of having a single role-independent representation of H which is pointed to from many different fields, we have many different "role-specific" representations. Activity in any one of these units then represents the conjunction of an identity and a role.

At first sight, the fully-parallel implementation seems very wasteful because it replicates the apparatus for representing and recognizing letters across all the different roles. However, the replication has some useful consequences. It makes it possible to recognize different instances of the same letter in parallel without any of the contention that would occur if several different processes needed to access a single, central store of knowledge simultaneously. Also, when letters are used as cues for words, it is not just the letter identities that are important. It is the conjunction of the identity and the spatial role within the word that is the real cue. So it is very convenient to have units that explictly represent such conjunctions.

In addition to the expense of replicating the recognition apparatus across all roles, the fully-parallel network has several other problems :

1. The replication raises the question of how, if at all, the multiple different role-specific representations of a given letter are related to one another.

2. As we go down the hierarchy, there are less and less units available for representing each constituent.

3. In a network which is not fully connected, it is not at all obvious how new knowledge can be incorporated without growing new connections.

For relatively shallow, man-made hierarchies of the kind that are important in reading or speech recognition, it may be tolerable to always devote less units to representing smaller fragments of the overall structure. But for domains like normal visual scenes this strategy will not work. A room, for example, may contain a wall, and the wall may contain a picture, and the picture may depict a room. We need to be able to devote just as much apparatus to representing the depicted room as the real one. Moreover, the very same knowledge that is applied in recognizing the real room needs to be applied in recognizing the room in the picture. If this knowledge is in the form of connections and if the knowledge is not duplicated there must be a way of mapping the depicted room into an activity pattern on the very same set of units as are used for representing the real room.

The distributed version of method 1

The Rumelhart and McClelland network uses localist representations in which each entity is represented by activity in a single unit. Localist representations are efficient if a significant fraction of the possible entities are present on any one occasion or if the knowledge associated with each entity has little in common with the knowledge associated with other, alternative entities. Both these conditions hold quite well at the level of letter recognition. For the more natural part-whole hierarchies that occur in everyday scenes, neither condition holds. Only a tiny fraction of the possible objects are present on any one occasion, so if one unit is devoted to each possible object almost all the units will be inactive. This is a very inefficient way to use the representational capacity. Also, different objects, like a cup and a mug, may have similar appearances and may make similar predictions. This means that there can be a lot of useful sharing of units and connections. Most of what we know about cups and mugs could be associated with a unit that is active for either a cup or a mug. If this method of sharing is taken to its logical conclusion we arrive at distributed representations in which each object is represented by activity in many units and each unit is involved in the representation of many objects (Hinton, McClelland, and Rumelhart, 1986).

One major advantage of using descriptions rather than single units as representations is that it is possible to create representations of novel objects (and also novel role-specific representations) by using novel combinations of the same set of primitive descriptors. This avoids the problem of having to find a suitably connected unit for each novel object. However, the other two difficulties of the fully parallel method are not solved by simply using distributed representations.

Method 2: Sharing recognition apparatus within a level

Distributed representations provide a way of sharing units and connections between alternative objects or alternative role-specific representations. In this respect they work just like pointers in a conventional computer memory. Instead of using a separate bit for each possible object that could be pointed to, each bit is shared between many possible alternative objects. As a result, a word of memory can only point to one object at a time. [3] The following analysis of the functions performed by a role-specific representation suggests a quite different and complementary method of sharing which can be used to share connectionist apparatus between the different role-specific instances that occur within one whole. In the Rumelhart and McClelland model each role-specific letter unit has three functions:

1. It recognizes the occurence of that letter in that spatial role. The recognition is accomplished by having appropriately weighted connections coming from units at the feature level.

2. It contributes to the recognition of words. This is accomplished by its connections to units at the word level.

3. Its activity level stores the results of letter recognition.

[3] Some ancient implementations of LISP actually use two separate role-specific representations within one word so that the first part of a word can point to one object and the second part can point to another.

There is an alternative model which uses role-specific letter units for functions 2 and 3, but not for function 1. Instead, it uses a single letter-recognition module which is applied to one position within the word at a time. Once the letter at the current position has been recognized, the combination of its identity and its position within the word activates a role-specific letter unit which acts as a temporary memory for the results of the recognition and also contributes to the recognition of the word (see figure 2).

The letter-recognition module must be applied to one letter at a time and so there must be extra "attentional" apparatus which selects out one portion of the parallel input (which contains features of all the letters), maps this portion into the input of the letter-recognition module, and also creates an explicit representation of where the currently selected letter lies within the word. Actually, the Rumelhart and McClelland model presupposes that there is apparatus of a similar kind in order to pick out the features of one word within a sentence or to cope with changes in the position of a word. So the new model does not require any qualitatively new attentional apparatus, it just requires it at the level of letters instead of at the level of words.

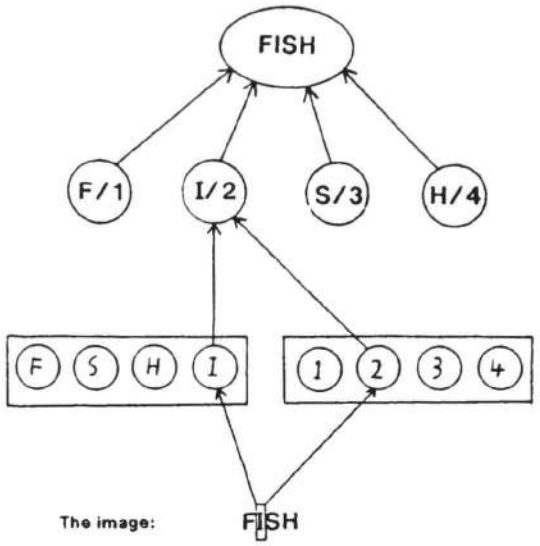

Figure 2: Some of the apparatus required to store the sequence of outputs of a single, sequential letter-recognition module in order to recognize a word. The network is "attending" to the second letter of the word. Notice that the role-specific units do not need to be able to recognize letters. The apparatus required for mapping the appropriate part of the input into the letter recognition module is not shown.

A connectionist dilemma

By sharing the recognition apparatus within a level, the network captures the regularity in the appearance of different instances of the same letter but it loses the ability to recognize all the letters of a word in parallel. This illustrates an important dilemma: If the knowledge is in the connections we can either capture the regularity by using a single module sequentially or we can have parallel recognition by using many modules at once. McClelland (1986) describes a clever but inefficient way out of this dilemma: There is a single central representation of the knowledge that is copied *during recognition* to produce the required parallel recognizers.

Figure 2 suggests an alternative way out. The serial recognizer could be used to train the parallel, role-specific recognizers. A network that time-shares a serial recognizer already requires role-specific units for storing its successive outputs. If these role-specific units had some connections to the perceptual input, they could learn to use these connections to predict the outcome of the serial recognition. The canonical, time-shared representation of the knowledge would then be acting as a superviser for the parallel recognition hardware. So the canonical knowledge would be transferred to the parallel recognition apparatus during learning which is much less demanding than transferring it during recognition. The hard part of learning is deciding what internal representations to use. Once this has been decided by the serial recognizer, it should be relatively easy to replicate this knowledge.

Method 3: Sharing across levels

There is one limitation of within-level sharing that is unimportant in the domain of reading but is very important in most other domains where the same knowledge can be applied at many different levels. For reading the knowledge is quite different at each level: Knowledge about the shape of a letter is quite different from knowledge about which sequences of letters make words, so there is little point in trying to use the same set of connections to encode both kinds of knowledge. In most natural domains, however, wholes and their parts have much in com-

mon. One example has already been given in which a room contains a picture that depicts a room. Another example is the sentence "Bill was annoyed that John disliked Mary." One of the constituents of this sentence "John disliked Mary" has a lot in common with the whole sentence. The same kind of knowledge is needed for understanding the constituent as is needed for understanding the whole. This is also typical of visual scenes which generally have just as much richness at every level.

If we consider how to map a part-whole hierarchy into a finite amount of parallel hardware there are three broad approaches:

1. The fully-parallel model uses a one-to-one mapping. Each object in the part-whole hierarchy is always mapped into the same set of units, and each set of units is always used to represent the same object.

2. Within-level sharing uses a many-to-one mapping. Many different objects at the same level can be mapped into the same set of units in the serial recognition apparatus. But whenever one of these objects is represented, it is represented in the same units.

3. Between-level sharing uses a many-to-many mapping. It allows many different objects at the same level to be mapped into the same set of units, but it also allows the same object to be mapped into different sets of units depending on the level at which attention is focussed.

The idea that the same type of object might be mapped to different sets of units is inherent in the idea of role-specific representations, but the idea that the very same instance can be represented in different ways depending of the focus of attention is a much more radical proposal. It is equivalent to viewing the hardware as a window that can be moved up and down (in discrete steps) over the part-whole hierarchy (see figure 3). One node in the hierarchy is chosen as the current whole and all of the units in the main network are then devoted to recognizing and representing this whole. Some units are used for describing the global properties of the whole, and others are used for role-specific descriptions of the major constituents of the whole. The entire pattern of activity will be called the "Gestalt" for the current whole.

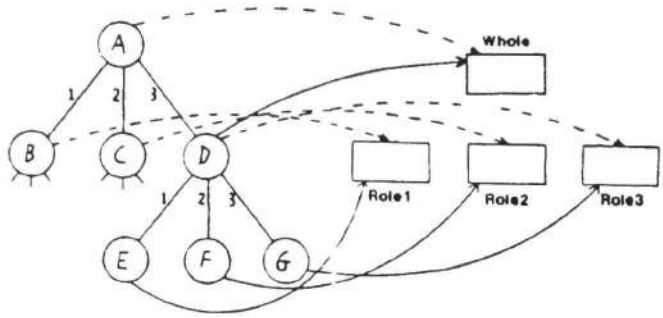

Figure 3: The solid and dashed lines show two different ways of mapping a part-whole hierarchy (on the left) into the same connectionist hardware (on the right). Notice that node D in the hierarchy can be represented by two totally different activity patterns that have nothing in common.

The crucial property of the moveable window scheme is that the pattern of activity that represents the current whole is totally different from the pattern of activity that represents the very same object when it is viewed as being a constituent of some other whole. In one case the representation occupies all of the main network and in the other case it is a role-specific description that occupies only the units devoted to that role.

The use of a many-to-many mapping raises many issues that do not arise or are not so important in the fixed mapping approach:

1. When the mapping between the world and the network is changed in such a way that one constituent of the previous whole becomes the new focus of attention, what kind of internal operations are required to convert the previous, role-specific description of that constituent into a full description that occupies the whole of the main network?

2. How is information about previous Gestalts stored so that the network can return to them later? The information cannot be stored as the activity pattern that the network settles to when the Gestalt is created because the very same network is needed for creating the next Gestalt.

3. How is the next mapping chosen?

There is not space here to address issue 3, but the following sections give a brief description of one way of handling issues 1 and 2.

Moving up and down the part-whole hierarchy

Figure 4 shows some of the extra apparatus that might be required to allow a connectionist network to move down the part-whole hierarchy by expanding a role-specific, reduced description into a full description of the role-filler. This corresponds to following a pointer in a conventional implementation. Notice that it is a slow and cumbersome process. Moving back up the hierarchy is more difficult. First, the full description of a part must be used to create the appropriate role-specific, reduced description of that part. This involves using the apparatus of figure 4 in the reverse direction. Then the role-specific, reduced description must be used to recreate the earlier full description of which it is a constituent. This requires some kind of content-addressable working memory for earlier Gestalts.

The obvious way to implement this working memory is to set aside a separate group of "working memory" units. If it is only necessary to remember one Gestalt at a time, this group can simply contain a copy of the pattern of activity in the network where Gestalts are formed. If several Gestalts need to be remembered at a time, several different groups could be used. Alternatively, a single group could be used provided that the various patterns of activity that need to be stored are first recoded in such a way that they can be superimposed without confusing them with one another. Examples of such encodings are described by Hinton (1981b) and Touretzky and Hinton (1985). Touretzky (1986) shows how this kind of working memory can be used to traverse and transform tree structures.

An interesting alternative implementation of working memory uses temporary modifications of the connection strengths in the network that is used for creating the Gestalt. Each internal connection in this network can be given two different weights: A long-term weight which changes relatively slowly and a short-term weight which is limited in magnitude, changes rapidly, and spontaneously decays towards zero. The effective connection strength at any time is simply the sum of the short-term and long-term weights. The long-term weights encode knowledge about which patterns of activity constitute good interpretations of the input to the network (i.e. familiar or plausible Gestalts). The short-term

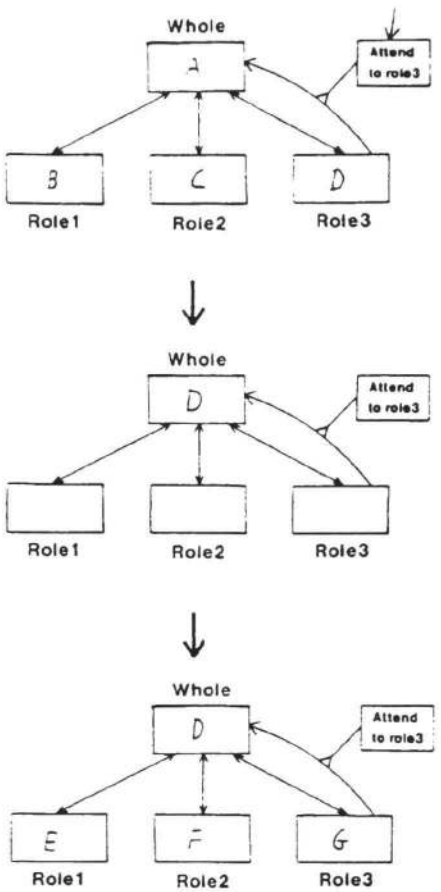

Figure 4: One way of using some additional hardware to allow the network to access the full description of node D from a role-specific reduced description. Even though the two descriptions correspond to quite different patterns of activity, their relationship should be non-arbitrary.

weights act as a contextual overlay [4] that encodes information about which patterns of activity occurred recently. If the network receives a rich external input which is incompatible with recently occurring Gestalts, it will settle to a new Gestalt and the short-term weights will act as noise (to which these networks are very resistant). If, however, parts of the external input are missing and the remainder fits some recently occuring Gestalt, the short-term weights will favor this Gestalt over other alternative Gestalts which would fit the partial input just

[4] Hinton and Plaut (1987) describe a very different use of this contextual overlay. It can be used to approximately cancel out recent changes in the long-term weights, thus allowing earlier memories to be "deblurred"

as well if the short-term weights were not considered. So the short-term weights will implement a content-addressable memory for recent Gestalts. Simulations (Hinton, 1973, unpublished) show that short-term weights can be used to allow the network to return to a partially completed higher-level procedure after executing a recursive call of the same procedure.

Conclusions

The combination of massively parallel constraint-satisfaction using reduced descriptions and relatively slow sequential access to full descriptions is a style of computation that is well-suited to networks of richly connected but rather slow processing elements. There is an inner loop of parallel, iterative processing in which the network performs a great deal of computation by settling into a state that satisfies constraints that are encoded in the connections. More elaborate computations which cannot be performed in a single settling are performed by a sequence of settlings, and after each settling the mapping between the world and the network may be changed. Changing the mapping corresponds to following a pointer (i.e. performing a remote access).

It is tempting to identify each change in the mapping between the world and the network with a single step in the network's "train of thought" This leads to an interesting view of what happens when a conscious cognitive process becomes automatic. Prolonged experience in a domain allows the network to develop reduced descriptions that make explicit the important regularities of the domain (see Hinton, 1986, for an example). This allows more of the computation to be done by interactions between the reduced descriptions, so there is less need to perform inherently sequential operations that change the way in which pieces of the task are mapped onto the parallel hardware.

References

W. D. Hillis. *The Connection Machine.* MIT Press, Cambridge, Mass., 1985.

G. E. Hinton. Learning distributed representations of concepts. In *Proc. Eighth Annual Conference of the Cognitive Science Society*, Amherst, Mass., 1986.

G. E. Hinton. Shape representation in parallel systems. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence, Vol 2*, Vancouver BC, Canada, 1981.

G. E. Hinton and K. J. Lang. Shape recognition and illusory conjunctions. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, 1985.

G. E. Hinton, J. L. McClelland, and D. E. Rumelhart. Distributed representations. In D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume I Foundations*, MIT Press, Cambridge, MA, 1986.

G. E. Hinton and D. C. Plaut. Using fast weights to deblur old memories. In *Proc. Ninth Annual Conference of the Cognitive Science Society*, Seattle, WA, 1987.

J. L. McClelland. The programmable blackboard model of reading. In J. L. McClelland, D. E. Rumelhart, and the PDP Research Group, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 2 Applications*, MIT Press, Cambridge, MA, 1986.

J. L. McClelland and D. E. Rumelhart. An interactive activation model of context effects in letter perception, part 1: an account of basic findings. *Psychological Review*, 88:375–407, 1981.

D. S. Touretzky. Reconciling connectionism with the recursive nature of stacks and trees. In *Proc. Eighth Annual Conference of the Cognitive Science Society*, Amherst, Mass., 1986.

D. S. Touretzky and G. E. Hinton. Symbols among the neurons: details of a connectionist inference architecture. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, 1985.