A Distributed Representation of Multiple Objects in a Visual Scene

Douglas Blank
Department of Computer Science
Indiana University, Bloomington, Indiana 47405
blank@cs.indiana.edu

1 The Problem of Grouping

Consider the problem of looking at a scene (such as the one shown in Figure 1), and mentally grouping, or structuring, the parts and pieces into a coherent view of the entire picture. Figure 1 could be "parsed" as three squares, two circles and a triangle. It could also be seen as two white and four black objects, as well as a myriad of other, different sets of groupings. These competing ways to view the scene can be affected by low-level attributes (e.g., *objects of the same color*, *objects that are on the bottom*, or *objects that are close to each other*). However one might also group objects using more conceptual categories (e.g., *objects that can roll*, or *objects that are of the type* square). Grouping is therefore sensitive to low-level perceptual pressures as well as those from high-level concepts [Hofstadter, 1984]. At first glance, our simple ability to group objects on these criteria might not seem to be of interest to the cognitive scientist. However, grouping appears to play a role in the formation of conceptual structures, and these structures are believed to be of central importance for many abstract mental processes, such as analogy-making [Gentner, 1983].

A system that is capable of grouping objects in scenes like the one in Figure 1 must be able to process perceptions, concepts, and multiple items. Symbolic systems have difficulty handling perception, and sub-symbolic (connectionist) systems have trouble handling multiple items simultaneously. However, there has been some success in recent years in the connectionist paradigm in processing multiple items.

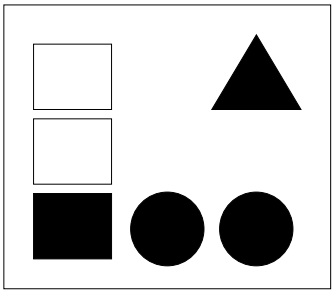

Figure 1: Given the task of mentally grouping these objects into sets, what groups might one form? People often form groups based on low-level, perceptual information as well as more abstract categorizations.

To simultaneously represent multiple items in a neural network, connectionists have developed the following options: 1) provide a separate bank of units for each item; 2) present the items as a sequence in time to a recurrent network; or 3) sequentially process each item as a role/filler pair. Options 1 and 2 are often hard to implement and do not generalize well, and 3 is useful only when the relationships (or the roles) of the items are known.[1] This paper describes a mechanism

[1] This category includes architectures such as Smolensky's Tensor Product formulation [Smolensky,

for representing multiple objects in a visual scene when the relationships between the objects have not yet been identified.

2 Taking *where* seriously

In representing visual objects in a scene, most neural network models will express *what* an item is as a pattern of activation in a particular bank of units. On the other hand most models "throw out" specific location information about *where* an object is seen. In fact, the very idea behind much of the connectionist visual research has been to generalize over position (see [Fukushima et al., 1983] for example). That is, the networks are designed to recognize an object *regardless* of where it is found. However, if we wish to create a viable model of object grouping, the *where* of each object must be represented as well as the *what*. In addition, not only must the location information be represented in the system, but it must also be bound to the *what* of the object. It is not enough to know that there is an object in the upper left-hand corner of the scene; the system must know that the object is a white square.

We have designed a representation which binds together the *what* and the *where* of multiple objects. This binding is based on Smolensky's Tensor Product (TP) formulation [Smolensky, 1990]. TPs are usually considered to be a mechanism for binding a role to a filler (e.g., the filler *Mary* to the role *Teacher*). TPs work by taking two vectors representing a role and a filler, and calculating the outer product (see Figure 2). This produces a matrix of values (activations) which represent the combination of that specific role, and that specific filler[2].

More importantly, TPs can also represent multiple role/filler pairs. Multiple role/filler pairs

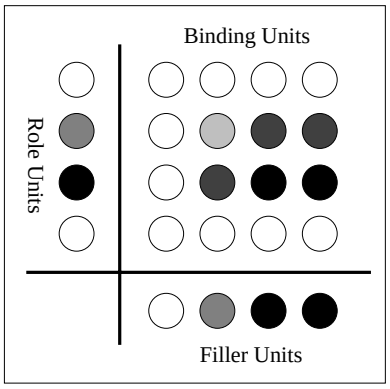

Figure 2: Smolensky's basic Tensor Product method. The outer product of a vector representing a filler and a vector representing its role is computed. The resulting matrix is a tensor of rank two representing the binding of the role to the filler. In these figures, the more activation at a unit the darker the color.

can be formed in the TP formulation by simply summing the individual binding matrices of each role/filler pair. For instance, the sentence *Mary gave Fred money* could be represented by assigning each word a unique role; we might bind together the following pairs: *Mary/giver*, *Fred/receiver*, and *money/object*. Then, by adding each corresponding unit's activation from each binding matrix, we have a representation for the entire phrase. Notice that since we are summing the binding matrices to form a new matrix, it is important that each role's representation be sufficiently orthogonal to the rest of the roles' representations. If this is not true, the newly formed matrix will have patterns which overlap by too much, and the bindings will become muddied.

A useful property of TPs is the ability to be easily *queried*. For instance, if we take a binding matrix representing a phrase and the vector representing the filler *Fred*, we can see what roles, if any, he played in the binding matrix. This is accomplished by taking the inner product (dot product) of the matrix and the vector. If we use the binding matrix from the above example, we would compute a vector of activations close to that of *receiver*, as that was what was bound to

1990] and Pollack's RAAM [Pollack, 1988].

[2]See [Smolensky, 1990] for a thorough examination of TPs and how they relate to other artificial neural network models.

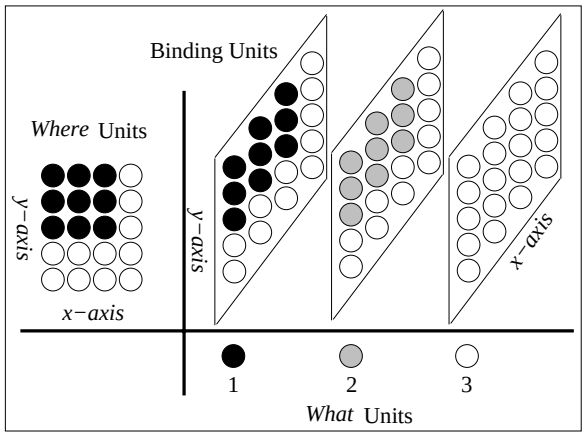

Figure 3: This variation of Smolensky's TP replaces *role* with *where*. The binding units can be thought of as maps, one for each unit in the *what* representation.

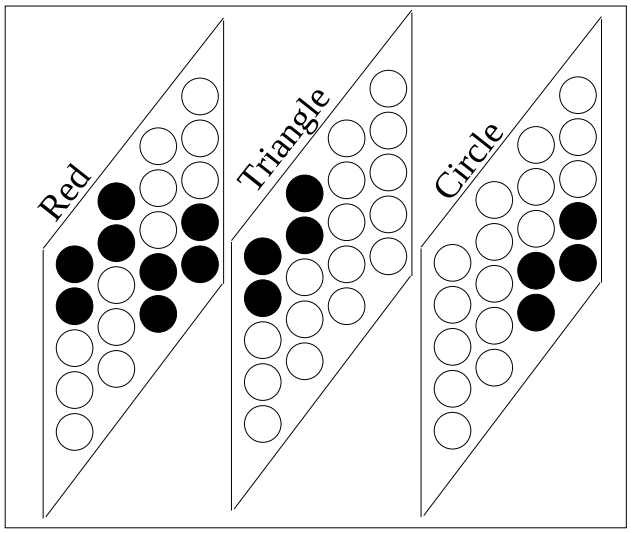

Figure 4: Encoding a scene with a red triangle in the upper left-hand corner and a red circle in the lower right-hand corner would produce a cube like that shown. Notice that all red objects can be located by simply looking at the red plane.

Fred.

The Tensor Product mechanism does allow for multiple objects to be stored in a single representation; however, it also requires that each object be paired with a role. In the case of object grouping in a scene of shapes, there are no such roles *a priori*. To apply the TP mechanism to representing objects in a scene, the vector which normally represents a specific role could be replaced by a pattern of activation which represents its *location* in the scene, as the roles are simply arbitrary vectors. Figure 3 shows a TP with the role vector replaced by a vector representing where the object is found in the scene. The location of the item is represented by activation; therefore, a larger object would have more units activated as it covers more area. In Figure 3, an object is seen in the upper left-hand corner of the scene which is then represented by activation on the units in the upper left-hand portion of the where-vector.

To bind the *what* to the *where*, the outer product of the where-vector and the what-vector is computed. As before, this results in a matrix representing the binding of the two vectors. However, since the where-vector is representing a two-dimensional image, one could think of the resulting matrix as a cube.[3] Each unit in the what-vector forms a plane, or map, of activation in the cube. For instance, unit 1 of the what-vector is multiplied by the where-vector and forms the plane of activations above unit 1 in Figure 3. If unit 1 represented *blue* then any unit that has activation on this plane indicates where a blue item can be found in the scene. The cube of an entire scene (i.e., multiple objects) is shown in Figure 4. The scene being represented is a red triangle in the upper left-hand corner and a red circle in the lower right-hand corner. The what-vector consists of a three unit, localist representation where units 1, 2, and 3 represent red, triangle, and circle respectively.

If the scenes are confined to non-overlapping objects, then it is guaranteed that the where-vector for each object's location is completely orthogonal to that of all other possible objects in the scene.

[3]Strictly speaking, the resulting cube is still a matrix, and, in TP terms, it is still a tensor of rank two.

3 A Network for Learning Analogies

To examine the usefulness of the what/where bound representations, we designed a network to perform a type of grouping task using these representations as input. The task was an analogy problem as follows: given a picture and a selected object from the picture, determine what the analogous object is in a second picture (see Figure 5.) For example, given the Target picture of Scene 1, we realize that the selected (underlined) object differs from the other two objects on the dimension of shape. If we then apply that difference to the Target picture, this would lead us to pick the black square of the Target picture because it too differs on the dimension of shape. Likewise, the underlined object of the Source picture in Scene 2 differs on the dimension of color, therefore, we might pick the white circle of the Target picture. This task apparently requires the network to compare the selected object with the other objects in the picture, categorize the difference, and map the difference onto the second picture. Notice that location plays no part in the analogy.[4] The network, therefore, was taught to perform analogies by being trained on similar analogies.

The network was a standard simple recurrent, back-propagation network (SRN).[5] It was trained in a two step process as follows:

- The Source picture is fed into the network:
  1. The Source picture's what-where matrix is computed and placed on the input layer
  2. The selected object is identified on another bank of inputs
  3. The source input is propagated, and the network is trained to identify the selected object
- The Target picture is fed into the network:
  1. The Target picture's what-where matrix is computed and placed on the input layer.
  2. The hidden layer from the previous time step is copied to the same bank of units that previously held the Source's selected object
  3. The target input is propagated, and the network is trained to identify the analogous object

We restricted our what-vector to represent only

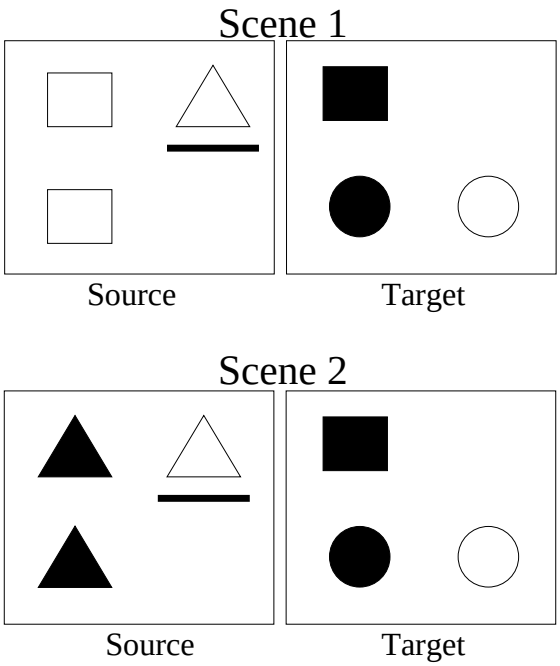

Figure 5: Here are two visual analogies. The goal is to find the object in the Target that is the "same" (in some analogical sense) as the underlined object in the Source. Scene 1 depicts a change-in-shape in the Source, therefore the analogous object might be the black square in the Target as it also differs from the others by a change-in-shape. Scene 2 shows a change-in-color, therefore the analogous object in the Target image would be the white circle.

[4] This is rather artificial; humans are influenced a great deal by location.

[5] Unfortunately, space limitations prevent a detailed examination of the network architecture here.

color and shape. The color was represented by a localist representation where one of three units was set to 1.0 and the other two set to 0.0. The three resulting orthogonal representations stood for red, green, and blue. Likewise, shape was represented by three units standing for square, circle, and triangle.

The where-vector was limited to a 7 by 7 retina. Additionally, only three shapes were placed in each scene, and their position was limited to one of the four corners. An object's location was represented by 4 units "on" (2 by 2) in the where-vector. The Source scenes were of the type shown in Figure 5. That is, they either had an object differ only by shape or had an object differ only by color. Given these constraints there were a possible 216 variations of the Source picture for each of the two types of analogies. The Target was generated as follows: create three objects all alike. Pick one object and change its shape, and pick a different object and change its color. There are a total of 1,080 possible Target pictures given these constraints. Therefore, there are a total of 466,560 unique combinations of Source and Target pictures. The network was trained on 9,850 of these.[6]

For each training step, a pair of the 9,850 Source and Target training pictures was chosen at random. Training proceeded as described above until the network scored 100% correct. This took approximately 68,000 training pairs. The network was then tested on a sample of the remaining 456,710 pairs (including novel Sources and Targets) and it successfully generalized over all of them.

[6] 10,000 sets of unique Source and Target pictures were generated for testing and training. Out of the 10,000, 150 were saved for testing, leaving 9,850 for training. Also, these were not a random 10,000 but rather a set specially selected to ensure generalization as opposed to mere memorization.

4 Conclusion

This research is part of an on-going project to model analogy. We believe that forming intermediate representations that lie somewhere between the perceptual and conceptual, and between the symbolic and subsymbolic will be the most beneficial for modeling complex human behaviors. In this paper, we have described and demonstrated the usefulness of a distributed representation of multiple objects in a visual scene. As this is the initial study of networks using this type of representation, many questions remain. However, the ability of the network to learn such abstract categories as "different color" and "different shape" suggests that this could be a useful representation in creating networks that exhibit more sophisticated abilities such as analogy-making.

References

[Fukushima et al., 1983] Fukushima, K., Miyake, S., and Ito, T. (1983). Neocognitron: a neural network model for a mechanism of visual pattern recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13:826–834.

[Gentner, 1983] Gentner, D. (1983). Structure-mapping: a theoretical framework for analogy. *Cognitive Science*, 7(2):155–170.

[Hofstadter, 1984] Hofstadter, D. (1984). The Copycat project: an experiment in nondeterminism and creative analogies. Technical Report AI memo #755, AI Laboratory, MIT, Cambridge, MA.

[Pollack, 1988] Pollack, J. B. (1988). Recursive auto-associative memory: Devising compositional distributed representations. In *Proceedings of the Tenth Annual Conference of the Cognitive Science Society*, pages 33–39. Lawrence Erlbaum Associates.

[Smolensky, 1990] Smolensky, P. (1990). Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence*, 46:159–216.