ANALOGICAL LEARNING AND INFERENCE IN OVERLAPPING NETWORKS

Paul Munro
pwm@pitt.edu
Yefei Peng
ypeng@yahoo-inc.com
School of Information Sciences, University of Pittsburgh,
Pittsburgh, PA 15260
USA

ABSTRACT

A method for training overlapping feedforward networks on analogous tasks is extended and analyzed. The learning dynamics of simultaneous (interlaced) training of similar tasks interact at the shared connections of the networks. The output of one network in response to a stimulus to the other network can be interpreted as an analogical inference. In a similar fashion, the networks can be explicitly trained to map specific items in one domain to specifc items in the other domain. The method has been applied to spatial tasks in a simple environments and to tree structures.

INTRODUCTION

Previous connectionist approaches to modeling analogical processing and/or skill transfer have been put forward, including:

- Explicit feature mapping (eg, Holyoak & Thagard, 1989; Halford et al., 1994; Hummel & Holyoak, 1997)
- Hybrid symbolic-connectionist models (eg, Mitchell, 1993)
- Resuse of weights from learning one task on a second task (eg, Pratt et al., 1991)
- Simultaneous training on multiple tasks (eg, Caruana, 1997)

It is well-known that feedforward networks trained by backpropagation develop internal representations that reflect both the simlarity structure of the input space and the space of desired output vectors (targets). In some architectures (eg, autoencoders), the internal representations reduce redundancy, while preserving essential information, such that the original pattern can be reconstructed.

Caruana (1997) showed that a network with multiple classification tasks with a common input can interfere in a constructive fashion. The approach here maps multiple input spaces to multiple output spaces, such that a common "abstract" representation space is formed in the internal layers of the system.

BACKGROUND

In a classic paper, Hinton (1986) demonstrated the role of internal representations in the network solution of a "family tree" task. In that paper, the network is trained on triples of the form <*agent, relation, patient*> to generate the *patient*, given the *agent* and *relation*. For example, given the input *agent*="Colin", and *relation*="mother", the network should compute an output that is a representation of Victoria (Colin's Mother). The network was trained to learn family relations from two disjoint family trees with identical structures; that is, there is a one-to-one mapping between individuals in the two domains. The hidden unit weights to the input layer are identical for many homologous pairs, and are exactly opposite for some others, since the output units are trained to distinguish between them. The network structure for this task is a precursor to the network described here; while Hinton's paper does not address analogy, similarities in the hidden unit representations of the homologues are apparent.

OVERLAPPING NETWORKS

The approach described here uses backprop in a network with multiple overlapping input-output pathways. Each pathway has a different training set. A simple example is depicted in Figure 1. Consider two learning domains, *A* and *B*, with training sets that may differ in their input and output representations; they may even be of different dimensionalities. The arrows $\boldsymbol{U_A}$, $\boldsymbol{U_B}$, $\boldsymbol{W}$, $\boldsymbol{V_A}$, and $\boldsymbol{V_B}$ represent complete connectivity matrices from one set of units to another, as indicated in the diagram. Thus, the weight parameters in $\boldsymbol{W}$ (as well as the bias parameters for units in H_1 and H_2) are shared and by both tasks.

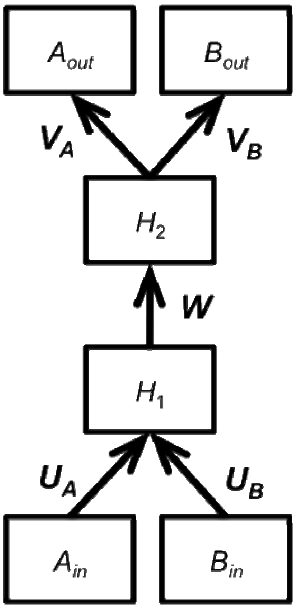

Figure 1. Two sets of input units, A_{in} and B_{in} project to a common hidden layer H_1, which in turn projects to a second hidden layer H_2, which projects to two output banks, A_{out} and B_{out} corresponding to the input banks. Arrows indicate full unit-to-unit connectivity between connected layers, labeled as $\boldsymbol{U_A}$, $\boldsymbol{U_B}$, $\boldsymbol{W}$, $\boldsymbol{V_A}$ and $\boldsymbol{V_B}$.

The subnetwork Net_A (A_{in}-$\boldsymbol{U_A}$-H_1-$\boldsymbol{W}$-H_2-$\boldsymbol{V_A}$-A_{out}) is trained using domain *A* for training with standard backprop. Similarly, the subnetwork Net_B (B_{in}-$\boldsymbol{U_B}$-H_1-$\boldsymbol{W}$-H_2-$\boldsymbol{V_B}$-B_{out}) is trained using domain *B* for training with standard backprop. Munro and Bao (2005) showed that first training one task to criterion can enhance training time on a second task. In the same paper, it is shown that interleaved training on two tasks produces a shared set of weights that is of still greater benefit to a related task. This supports the conjecture that the shared weights are more likely to encode common features under interleaved training, than they are by completing one training regimen before starting the second.

An overlapping network trained by interleaving backprop trials on Net_A with backprop trials on Net_B can be analyzed for its structural mapping properties by simple examination of B_{out} generated by A_{in} stimulation. Munro (2008) demonstrated such “crossactivation“ of overlapping networks trained on similarly structured tasks. In this example (reprinted from Munro, 2008), the task is to generate the “neighborhood” of a point in a 4x4 grid. There are 16 input units and 16 output units. In each input pattern, just a single unit is activated (the other 15 are silent), and the associated target from the training set is the set of units in its immediate “neighborhood”. Figure 2 (top) shows the output from a successfully trained network. Each array shows the output acivities generated by stimulating the input unit in a particular position. For example, the array in the upper left shows activity in the upper left unit and its two neighbors – the neighborhood of the upper left point on the array. The bottom array in Figure 2 shows the output generated at B_{out} by a stimulus from A_{in}. Note that the output generated by the lower right corner generates the output than would have been generated by the *upper* right corner of B_{in}. By visual inspection, it is evident that the network has mapped the stucturally identical tasks with a vertical inversion, which is consistent since these tasks have vertical symmetry (more precisely, 4-way symmetry).

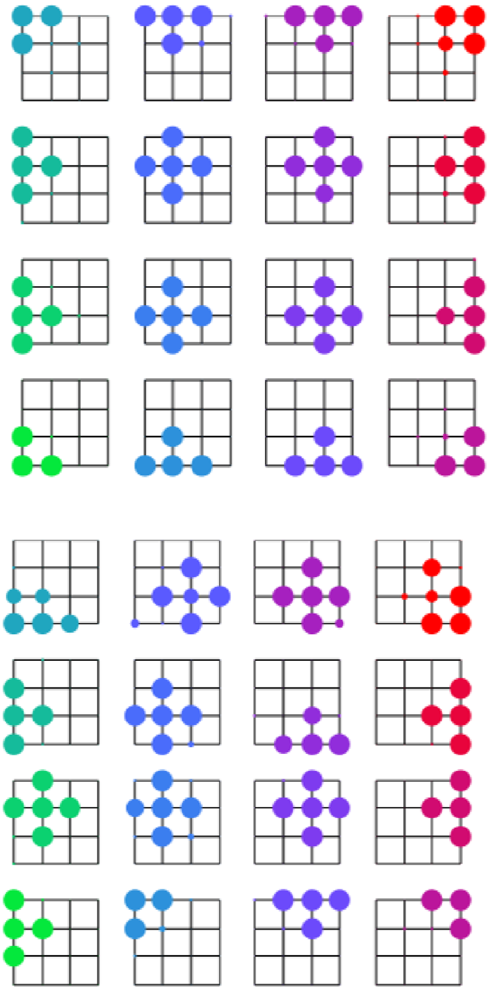

Figure 2. Top: A 4x4 array of 4x4 arrays. Each (small) array depicts the output pattern generated by a single input unit. Bottom: Output from crossactivation of B_{out} by A_{in}.

A potential problem in structural mapping is that of ambiguity resolution. It is possible, and for some tasks likely, that the mapping of a paricular item is ambiguous or nearly ambiguous.

CROSS TRAINING

Just as cross activation is a useful tool for analyzing mappings inferred by interleaved training, a "cross-training" procedure can be used to enforce the mapping of a specific item from domain A to a specific item in domain B. Crosstraining a stimulus from A_{in} to a desired target in B_{out} is achieved by running the backprop procedure on the crossed subnetwork Net_{AB} (A_{in}-$\mathbf{U_A}$-H_1-$\mathbf{W}$-H_2-$\mathbf{V_B}$-B_{out}).

Not all pairs are needed for crosstraining. Presumeably, the system can be given "guidance" by providing some known homologues for crosstraining.

Crosstraining introduces two training sets in addition to A and B. Let X_{AB} and X_{BA} be the sets of A_{in}-B_{out} pairs and A_{out}-B_{in} pairs respectively. Each of the four training sets corresponds to one of the four subnetworks Net_A, Net_B, Net_{AB}, and Net_{BA}.

The relative frequency with which each of the four subnetworks is trained is an important consideration, especially the amount of crosstraining relative to "vertical" training.

Figure 3 shows the same 4x4 neighborhood task with crosstraining. The four corner stimuli and the four central stimuli in the A domina have been crosstrained to the items in B that are rotated 90 degrees clockwise. For these 8 items the rotated patterns map precisely. Even though the remaining 8 patterns are not crosstrained, it is evident that they are crossactivating in an appropriate fashion.

Vertical training (A_{in}-A_{out} and B_{in}-B_{out}) can yield analogical corresponences that can be examined by cross activation (as seen in Figure 2). These correspondences constitute a structural mapping that is "discovered" by the system. Inclusion of crosstraining examples in the training procedure results in structural mappings that are much clearer, and seem likely to be more robust. Vertical training is thus an "unsupervised" form of learning analogies, whereas cross-training is "supervised"; in effect, the crossactivation of items not in the crosstrained sets is a form of analogical inference.

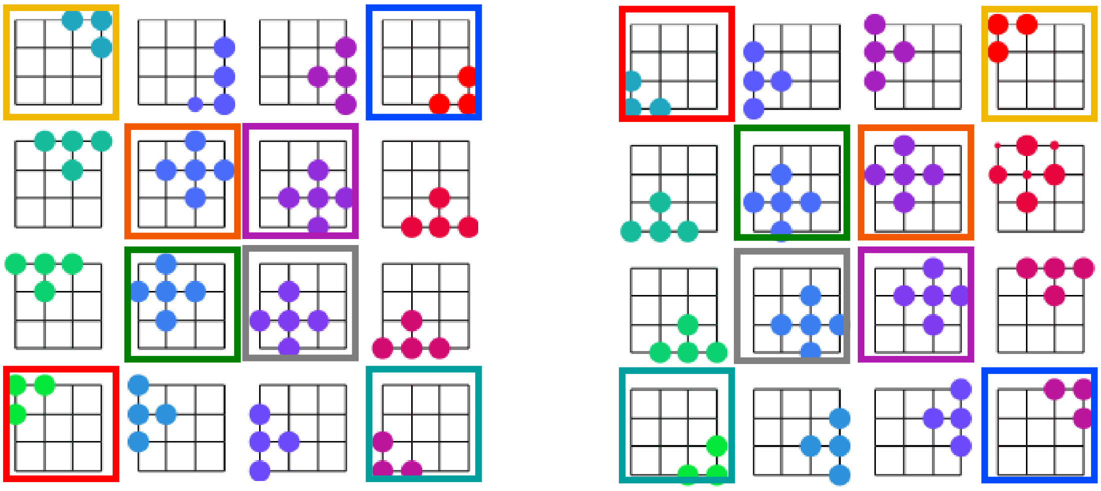

Figure 3. Crossactivation of A-B (left) and B-A (right) after training with interleaved crosstraining . Colored boxes show the crosstrained pattern pairs. The crossrained pairs show excellent mapping and the inferred crossactivations (i.e. the patterns not crosstrained) behave as expected, for the most part. For example, the activation pattern in the right array in the second row and the rightmost column is not correct.

NON-IDENTICAL TASKS

Since the networks overlap only in the "deep" layer(s) of the network, the surface representations of the stimuli (i.e., the input and output patterns) need not be of the same dimension. In principle, the suface represntations of the two tasks may be entirely different. The representations of the two tasks at the first hidden layer are in a common space, which provides input to the downstream layers of the network. The simulation result in Figure 4 illustrates network learning of a 3x3 environment simultaneously with a 5x5 environment. The upper left panes (labeled **A-A**) shows the output generated by the nine units in A_{out} for each of the nine input units. Note that this vertical task is learned virtually perfectly – each input unit activates its neighbors and no other units. Similarly the 5x5 vertical task is learned, as can be seen visually (**B-B**).

Here, all of the stimuli in the 3x3 envorments have crosstrained to corresponding points in the 5x5 environments. The colored squares indicate the cross-training examples. Note that, as in the previous example, the non-crosstrained patterns in the B-A plot (lower right) seem to interpolate between the patterns that are cross-trained in a quasi-sytematic fashion. In some cases, the response properties virtually replicate those of an adjacent input -- compare the 3x3 grid in the upper left (yellow square) with the grid immediately to the right. In other cases, the response grid reflects a blend of the adjacent cross-trained responses.

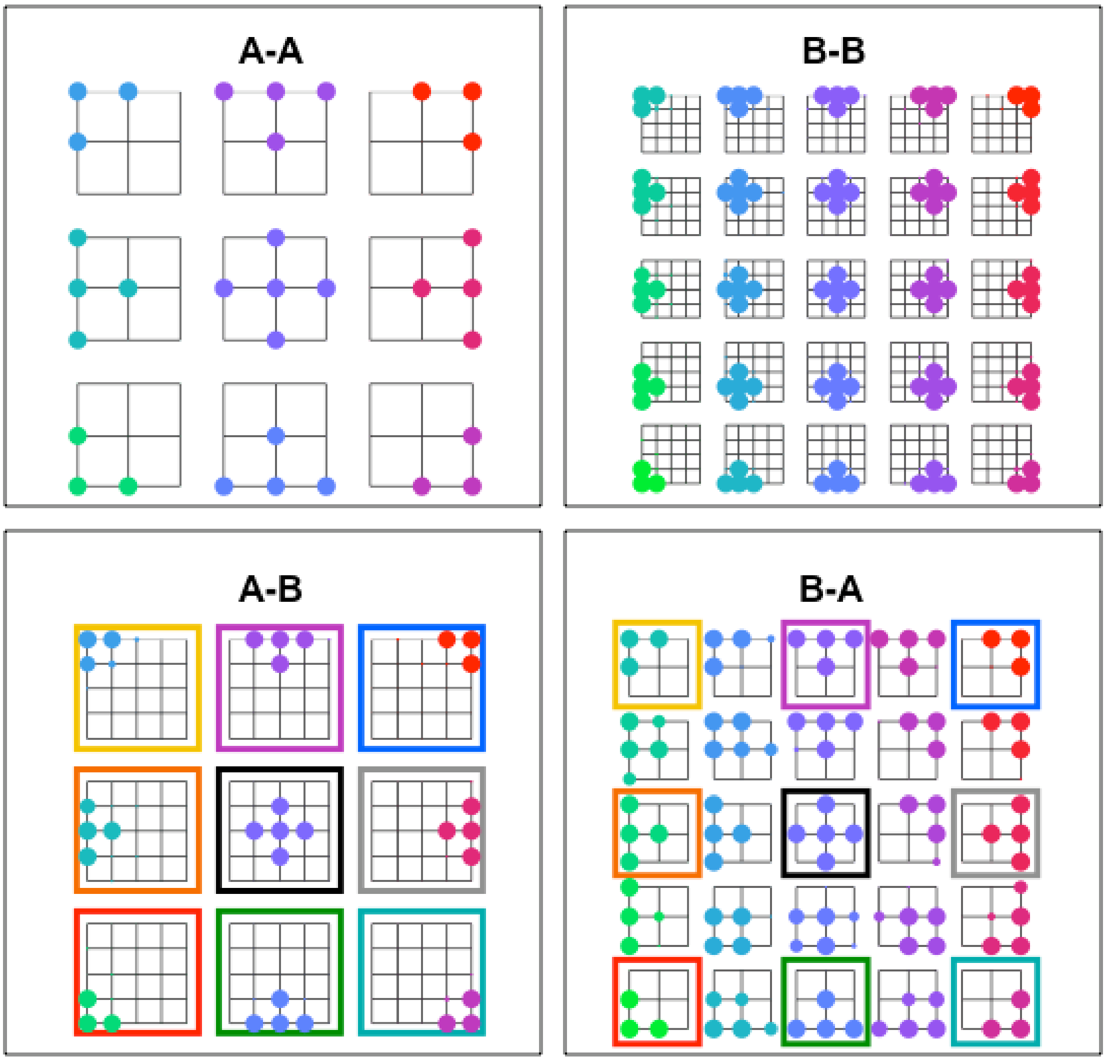

Figure 4. A network trained on neighborhood tasks in a 3x3 grid and a 5x5 grid (see text for description).

ONTOLOGY MAPPING

A possible use for overlapping networks is in the area of ontology mapping. The problem is to find correspondences between items in one ontology and another, usually in the same domain. For example, two different hospitals may have developed ontologies independently and wish to merge their databases. Overlapping networks with cross-training can be used to approach problems in ontology mapping. An ontology can be represented as a graph with one or more types of arcs.

A network can be trained to learn a tree structure using neighborhood relationships like those in the examples above. The pair of graphs in Figure 5 illustrate a very small graph matching problem. The trees do not match exactly and certain pairs are given: (r,R), (a,A), etc. That is, in addition to training the network on each tree vertically, the network is cross-trained on these pairs. The numerically labeled nodes are left for the system to work out. Analysis of several runs shows a small number of solutions that there are generally a few solutions, most of which seem “sensible”.

DISCUSSION

When a connectionist network is trained (for example, with backprop), the connection strengths encode statistical properties of the joint distribution of the pattern pairs in the training environment. Unlike most models trained with backprop, it is important to note that the various weight matrices in this network architecture are *not following gradients on the same error surface*. Let the errors be defined as E_A and E_B (the precise form of the function is not important for this discussion), and let χ be the cross-training coefficient. The weight matrices $\boldsymbol{U_A}$ and $\boldsymbol{U_B}$ are following gradients in the “pure errors” E_A and E_B respectively, while $\boldsymbol{W}$ is being modified to minimize the combined error $E_A + E_B$. Finally, the lower weights $\boldsymbol{V_A}$ and $\boldsymbol{V_B}$ are following gradients in a combined error such as $E_A + \chi E_B$.

It seems reasonable to suggest that, to this extent, the neural processes underlying analogical processing make use of overlapping pathways. An obvious candidate is the prefrontal cortex.

This approach to structural analogies and ontology mapping bears further study. As in Hinton’s original work, the inclusion of differnet kinds of relationships is desireable

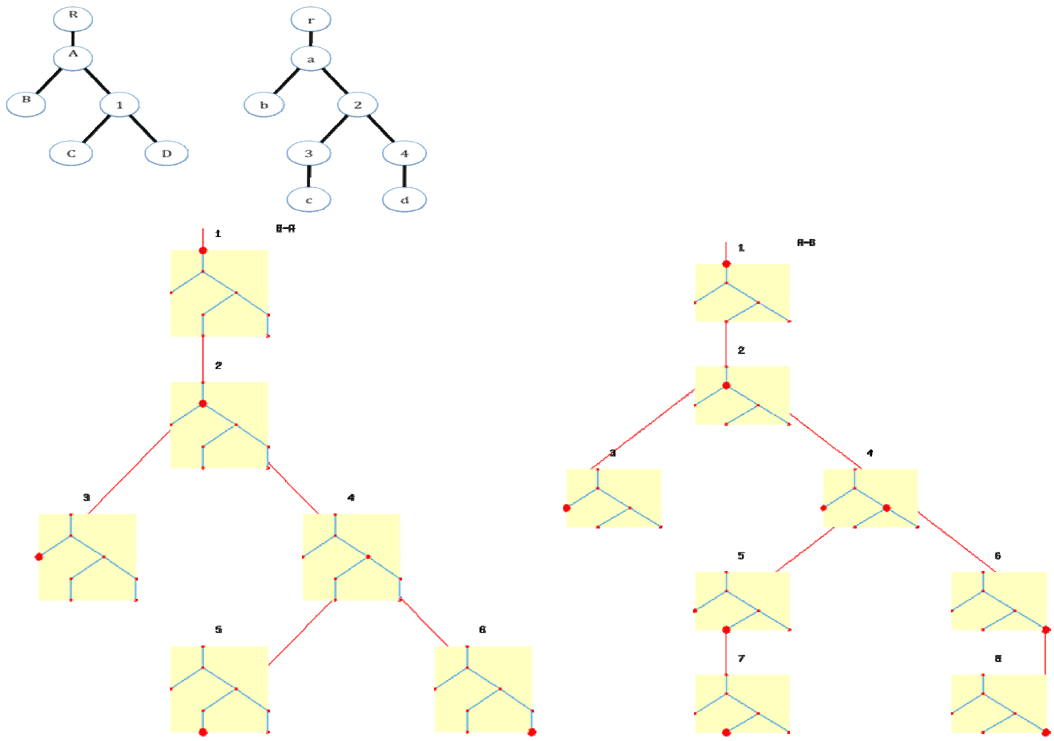

Figure 5. A graph-matching problem to illustrate ontology mapping. Top: A pair of graphs. Upper and lower case labels indicate corresponding elements in the two graphs. Mapping of the numerically labeled

nodes is to be inferred by the system. Bottom: Simulation results shows no inference for node 1 on the left graph. Inferences for the right graph are 2→1, 3→C, and 4→D.

References

Caruana, R. (1997) Multitask learning. *Machine Learning, 28*:41-75.

Gentner, D., (1983) Structure-mapping: A theoretical framework for analogy, *Cognitive Science 7*:155-170.

Ghiselli-Crippa, T. & Munro, P. (1994) Emergence of global structure from local associations. In: J. D. Cowan, G. Tesauro, J. Alspector (eds.) *Advances in Neural Information Processing Systems 6.* San Mateo, CA: Morgan Kaufmann Publishers.

Halford, G., Wilson, W., Guo, J., Gayler, R., Wiles, J., Stewart, J. (1994). Connectionist implications for processing capacity limitations in analogies. In: K. Holyoak & J. Barnden (eds.) *Advances in connectionist and neural computation theory, vol. 2, Analogical Connections,* pp. 363--415. Norwood, NJ: Ablex.

Hinton, G. (1986). Learning distributed representations of concepts. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society,* pages 1-12, Amherst, Lawrence Erlbaum, Hillsdale.

Holyoak, K. & Thagard, P. (1989) Analogical mapping by constrant satisfaction. *Cognitive Science 13,* 295-355.

Hummel, J. & Holyoak, K. (1997) Distributed representations of structure: A theory of analogical access and mapping. *Psychological Review, 104,* 427- 466.

Mitchell, M. (1993) *Analogy-making as Perception: A computer model*. Cambridge, MA: MIT Press.

P. Munro and J. Bao (2005) A connectionist implementation of identical elements. *Twenty Seventh Ann. Conf. Cognitive Science Society Proceedings*. Lawerence Erlbaum: Mahwah NJ

Pratt, L. Y., Mostow, J., and Kamm, C. A. (1991) Direct transfer of learned information among neural networks. In: *Proceedings of the Ninth National Conference on Artificial Intelligence* (*AAAI-91*) Anaheim CA

Thorndike, E. L.., & Woodworth, R. S. (1901). The influence of improvement in one mental function upon the efficiency of other functions. *Psychological Review, 8*, 247-261.