Connectionist modeling of part–whole analogy learning

Peter Gergeľ (peter.gergel@gmail.com)
Faculty of Mathematics, Physics and Informatics, Comenius University
Mlynská dolina, 84248 Bratislava, Slovak Republic

Igor Farkaš (farkas@fmph.uniba.sk)
Faculty of Mathematics, Physics and Informatics, Comenius University
Mlynská dolina, 84248 Bratislava, Slovak Republic

Abstract

Analogical reasoning, along with inductive, deductive or abductive reasoning, belongs to the fundamental human mechanisms for the environment exploration, learning, or problem solving. Modeling this ability using computer simulations is important, as it might offer mechanistic explanation of these phenomena. In this work, we focus on the part–whole analogies in a separation task where the analogical objects between two scenes show a mutual resemblance. In simulations, using a simple recurrent network, we deal with the problem of geometrical analogies, inspired by the Analogator model (Blank, 1997). The original model had learning limitations hindering its full potential, combined with increased time and memory complexity. We propose model modifications for removing these limitations which leads to superior learning performance both in terms of speed and accuracy.
Keywords: geometrical analogies; part–whole relationship; simple recurrent network; learning

Introduction

Analogical reasoning is considered to be the fundamental cognitive ability, which distinguishes humans from other animals.[1] This mechanism enables to explain new concepts in terms of old ones, to emphasize some aspects of situations, to generalize, to characterize situations, to explain or describe some new phenomena, to provide ideas on how to behave in new circumstances, to understand new forms of humour, etc.

To create an analogy means to see an object or situation within one context as the same as different object or situation within another context. Hall (1989) defines this process as a mapping between *source* domain which represents old, known situation onto *target* domain which represents new, unknown situation. Objects between domains that possess the same properties are mapped onto each other. Whole process consists of four steps: (1) source identification, (2) evaluation of degree of similarity, (3) information transfer from source to target, (4) consolidation. That is, when one starts making an analogy, the first step is to recall a similar situation from the past that resembles the current situation (source identification). Then one evaluates whether the resemblance is adequate (degree of similarity) and if so, one tries to use the knowledge about the past situation in the new situation (information transfer). The outcome of this process is then remembered (consolidation).

There are many types of analogical reasoning with different histories. A classical type of analogy is considered the *proportional analogy* "A is to B as C is to D", that dates back to ancient Greece. For example, "white is to black as cold is to hot", or "tree is to forest as department is to faculty". The next type we could consider is a *metaphor* which is a figure of speech that describes one object in terms of a second object: e.g. "He has a heart of stone".

In our paper, we focus on *part–whole* proportional analogies where analogical objects (parts) between the scenes (wholes) are the ones that share a common attribute; this could be the same role, position, the same or very similar structure, or similar sensory properties. One type of analogies in this category are geometrical analogies in which two scenes comprising geometrical objects are presented and one scene contains the selected object. The idea is to select an object in the second scene which is analogical to the selected object in the first scene in some way. The concept of learning and analogy making in people is still poorly understood, despite plethora of existing computational models (Gentner and Forbus, 2011). Therefore, cognitive modelling is important as it may provide new hypotheses and explanations.

This type of geometrical analogy was dealt with by Evans (1968) using symbolic approaches and more recently by Lovett et al. (2009) who combined the sketch understanding program CogSketch (Forbus et al., 2008) and the symbolic model SME (Falkenhainer et al., 1986). Connectionist solution to these part–whole analogies was proposed by Blank (1997) whose doctoral thesis served as the main inspiration for our work. More recent connectionist models have not dealt with this type of analogy. Blank designed *Analogator* model as a simple recurrent network that learns to reason analogically by seeing lots of analogical pairs. This represents quite a different approach, as many other models were explicitly designed for analogical reasoning and they did not have to learn. However, *Analogator* had its limits which prevented it from using its whole potential.

Geometrical analogies

Consider two scenes consisting of 3 objects with different shapes and colors. In the first scene (*source*) exactly one object is selected (figure) which differs in some way from the remaining two (ground).[2] The idea is to identify this unique property and to choose an object in the second scene (*target*) that differs

[1] However, the evidence suggests that chimpanzees are also capable of analogical thinking to some degree (Gillan et al., 1981).

[2] We use the terms "figure" and "ground", even though this task does not evoke the experience of perceiving the figure as closer to the observer than the ground. Blank (1997) also used these terms in his model that we were inspired by.

from the other objects in the same way as the selected object in the source.

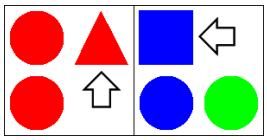

Figure 1: Example of a geometrical analogy.

Example of this problem is given in Figure 1 where on the left (source), one object is beforehand selected (see the pointing arrow). On the right (target), the correct answer is also shown (blue square). We have constrained the problem to three different colors (red, blue, green) and three different shapes (circle, square, triangle). Hence, every scene consists of exactly three objects, which are placed in three possible positions (out of four corners). In Figure 1, red triangle is analogical to blue square because both differ in shape from the ground. Another possible example could be the pair of pictures where analogical objects differ in color, or where the target is the rotation of the source.

We have chosen two types of analogies to investigate (following Blank): rotational (target is the rotated version of the source) and categorical (analogical objects differ in color or shape from the ground).

Rotational analogies In this type of analogy the target scene is made by rotation of source scene by 90°, 180°, 270° or 0° (no rotation applied) and changing the shape and color of objects. In Figure 2, blue triangle bottom left is analogical to red square top right, assuming the rotation 180°. Color and shapes are in this type of analogy not considered because they could create ambiguity.

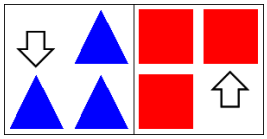

Figure 2: Example of a rotational analogy.

Let us now evaluate the overall number of unique rotational analogies. Because all objects on the scene have the same color and shape, there are exactly 9 possible options for choosing one object (3 colors $\times$ 3 shapes). Then there are 4 possible positions on the setting of objects in the scene[3] and 3 possible selections for the figure object. Therefore there are $9\times4\times3 = 108$ unique source scenes and for every scene there are $9\times4 = 36$ target scenes. To sum up, we have $108\times36 = 3888$ unique pairs of two scenes between which it is possible to find a rotational analogy.[4]

[3]The number of all subsets of size 3 (3 objects) over the set of size 4 (4 options for putting objects in a scene) sums up to $\binom{4}{3} = 4$.

[4]Blank actually generated 46656 unique pairs because he did not constrain his experiments to using only one color and one shape in a scene. We did so because we would like to use both rotational and categorical analogies as an input for the same model, and we needed to avoid ambiguity in analogy making.

Categorical analogies In this case the analogical object differs from the ground by color or by shape. In the source scene the selected object differs by one property from the ground, either in color or in shape while in the target scene it differs in both properties. The reason for that is that we want the network to notice this unique property in the source scene and then find it in the target scene. The network should learn to identify the required property and to ignore the rest. Example of this categorical analogy in which analogical objects differ in color is in Figure 3. Example of a categorical analogy where objects differ by shape is shown in Figure 1.

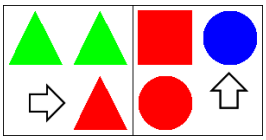

Figure 3: Example of a categorical analogy where analogical objects differ in color.

Scene representation

In order to approach the task, we must also choose the data representation for inputs and targets. Basically, there are two possible ways how to represent data: implicitly or explicitly. Both are explained in the following sections.

Implicit representation When using implicit representation, the structure of objects and their relations to others are not explicitly defined. Instead, this representation only defines attributes of objects; it does not provide information about relationships between attributes, neither which attributes belong to which objects. For example, when a geometrical object is not defined by the set of points, but instead by the set of colors in the bitmap. Motivation behind using implicit representation is that in real world there are probably no explicit representations, since every percept is obtained implicitly. In order to create an analogy, one has to extract required attributes and to map them in a way that analogy is created. If the input was explicit, extraction of attributes would no longer be required and the problem of analogy creation would be much simplified. Mitchell and Hofstadter (1995) state that perception of a situation is an important component in the process of creating analogies and it should be included in the computational models for analogy making. The fact that perception is a part of analogy making was one of the most important contributions of cognitive science in understanding analogy making in humans.

Blank was inspired by the work of Halford et al. (1994) who used *tensor representation* to implicitly represent predicates and arguments.[5] Blank represented the scene with geometrical

[5]It is questionable whether this type of implicit representation is cognitively plausible, though.

objects as follows: For every object he created a 2×2 matrix **X** which defined the position of an object in the scene and a vector **y** of length 6 to define object attributes. By using the outer product (**X** ⊗ **y**) he obtained a tensor of order 3 (6 matrices). Example of representing red square in top right corner can be seen in Figure 4.

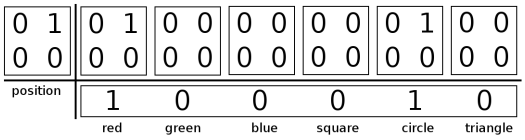

Figure 4: Tensor representation for red square top right.

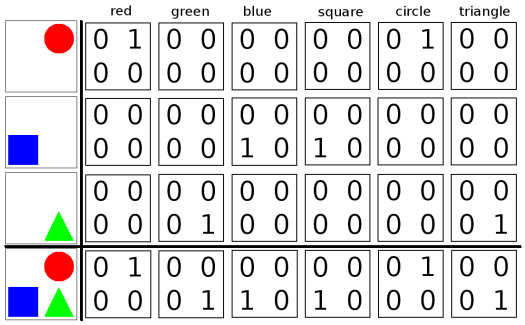

Figure 5: Tensor representation of the whole scene.

To get the tensor representation for the whole scene, Blank generated tensor representations for each object and then summed up the corresponding matrices. Detailed example is shown in Figure 5. Furthermore, it is also necessary to represent the selected object. This can be simply done by providing another 2×2 matrix which defines the position of this object. To sum up, the whole input consists of tensor representation of the source scene, of the position of a selected object and also of the target scene.

Explicit representation Symbolic models of analogy making typically use this approach which led to their criticism (Chalmers et al., 1992). We already mentioned that using explicit representation in analogy making simplifies this process to a large degree. Another problem with explicit representation is that there are many unique and equivalent means of representing the scenes, but not all of them would allow analogy to be correctly formed. The reason why we decided to also use explicit representation is to compare them with implicit representation.

Our exlicit representation abstracts from the problem of feature binding, which implies that all features are already linked to corresponding objects. It uses boolean vector of length 6 for description of one object. Every element of this vector defines which attribute is present. The first three bits define the color and the last three bits define the shape. Figure 6 provides an example.

To represent the whole scene, vector representation is created for each object and all vectors are simply concatenated to form one vector of length 6×4 = 24. Example of this representation is shown in Figure 7. Position of a selected object is defined the same as in implicit representation (Figure 4).

Figure 6: Explicit representation of green square.

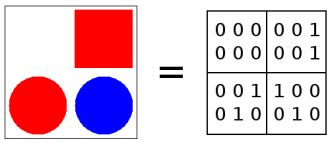

Figure 7: Explicit representation of the whole scene.

Analogator model

Connectionist model *Analogator* (Blank, 1997) is a domain-independent model that learns part–whole analogies based on examples. The basic difference between *Analogator* and other models is that *Analogator* was not explicitly designed to make analogies. It is based on a simple recurrent network (Elman, 1990). The input layer is divided into two groups of neurons: the first group contains representation of the whole scene, while the second group (context neurons) stores the selected object, or a copy of the hidden layer activation. The output layer also comprises two groups of units where the first group contains the position of the selected object and the second group outputs the remaining two objects at correct positions (see Figure 9).

Computation related to processing a source–target pair of inputs is executed in two steps (see Figure 8): In step 1, the network learns to properly separate in the source scene the figure (selected object) from the ground (the remaining two objects). The source scene with a selected object is presented to the network and the output layer activations are calculated (step 1a). Afterwards, the correct targets (figure–ground pair) are placed at the output and the weights are trained (step 1b). In step 2, the model learns to apply the previous transformation (from step 1) to the target scene. Here the target scene is placed at the input, the hidden layer activations from previous step are copied into context neurons and the outputs are calculated (step 2a). The network weight are then updated to produce analogical figure–ground pair at the output layers (step 2b).

The limitation of this architecture is that the number of neurons in the hidden layer and the context part must be identical. Because the context part also stores the position of a selected object (dual purpose layer), four neurons (there are 4 places the selected object) may be too restrictive for the hidden layer. On the other hand, increasing the number of neurons would introduce redundant neurons in representing the position of a selected object.

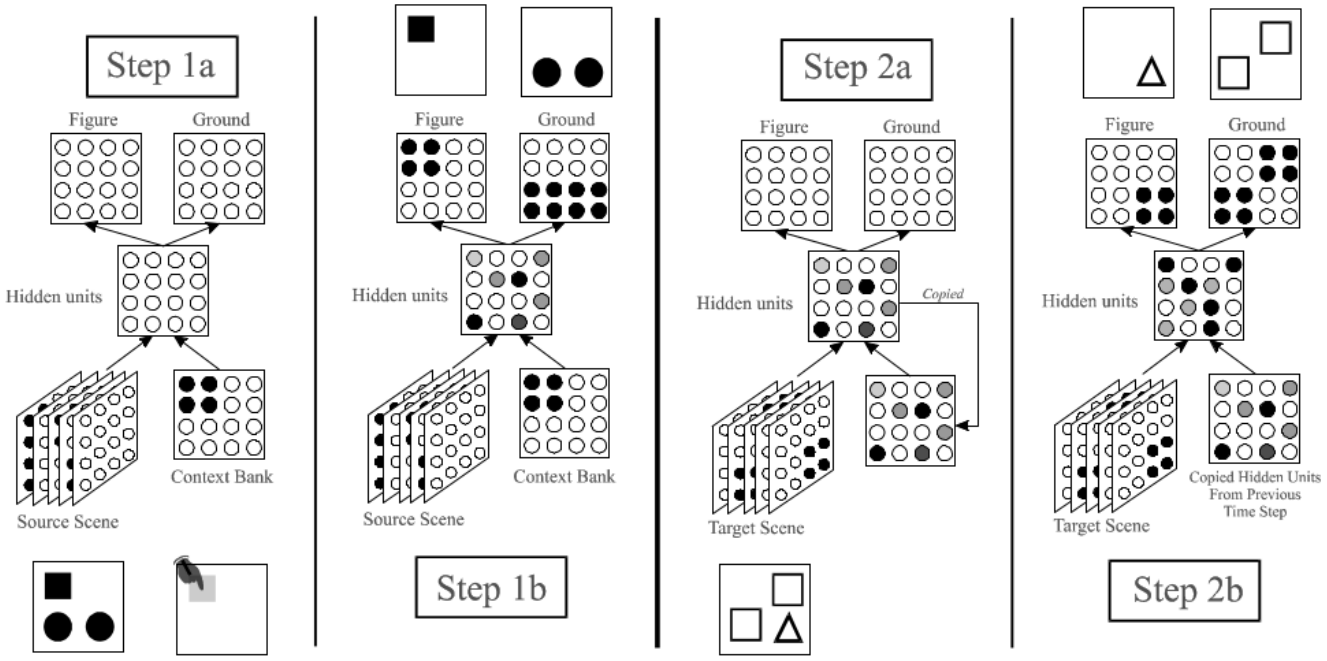

Figure 8: Analogy making between the source scene and the target scene (Blank, 1997). For explanation, see the text.

Modifications of the basic model

We have proposed and tested several modifications of the basic model which eliminated its restrictions. First, we split the input layer into three groups of neurons instead of two. The first group remained unchanged (the whole scene), while the second group was split to separately represent the position of the selected object (in step 1) and the context. In step 1 the context neurons are set to zeros (they are not required at that time) and in step 2 the hidden layer is copied into context neurons while the neurons storing the position of a selected object (the second group) are set to zero.

Second, we applied linear transformation to hidden layer activations via Principal Component Analysis to project the patterns into a lower dimension and stored its result into neurons representing the position for a selected object. In this model, PCA layer served as the layer of context neurons. To implement online PCA learning, we used the Generalized Hebbian learning algorithm (GHA, Sanger 1989). Model with this modification is trained simultaneously with BP and GHA algorithms.

Third, we introduced the second hidden layer to the model assuming it would increase its ability to learn analogies, speed up convergence and improve generalization. Combination of these three modifications yielded a variety of models as shown in Table 1.[6] Architectures of these novel models can also be retrieved from Figure 9.

[6]Combination of PCA and the split input layer would not make much sense, since both attempt to solve the same problem.

Training the models All models were trained on both types of analogy (rotational and categorical, having two subtypes). All weights were initialized to random values with uniform distribution $(-0.05, 0.05)$. In each step, the standard error backpropagation (BP) algorithm was applied. Each epoch consisted of several thousands of input–target patterns (i.e. pairs of a source and a target scene). For interpretation of the network outcome, the values were rounded to 1/0 (with the threshold 0.5). These values were afterwards compared with desired outputs to calculate the error. Learning was stopped after the fixed number of epochs.

Table 1: Overviews of the tested models. Each model could be combined with explicit or implicit representations (B = basic, S = split, P = PCA, 2 = extra hidden layer).

Model	PCA	split inp.	hid2
AB1			
AP1	✓		
AS1		✓	
AB2			✓
AP2	✓		✓
AS2		✓	✓

Experiments

Because training the model on one type of analogy (either rotational or categorical) is an easy task (it was demonstrated by Blank) we decided to simultaneously train our models on both types of analogy. In order to analyse the impact of the

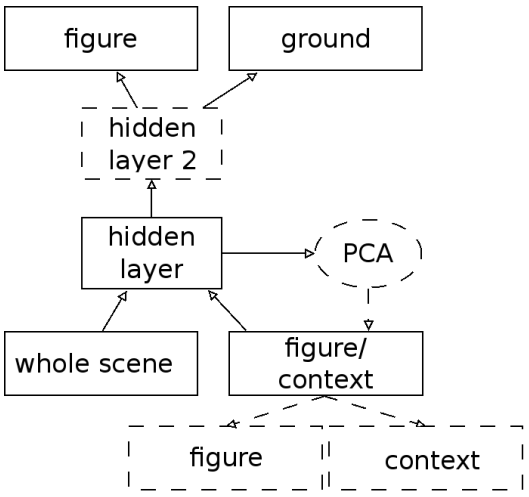

Figure 9: Proposed model modifications. The original model (Analogator) consists of solid boxes, novel elements are illustrated by dashed boxes.

modifications we also trained the basic models with both types of representation to be used as a reference. We trained each model five times to get an estimate of average training and testing errors. Every simulation used different patterns for training and testing sets but lasted the same number of epochs. The set of patterns was generated as follows: First, we generated all categorical analogies from which we randomly selected a subset of 20000 patterns. Then we extended this set by all (3888) rotational analogies which yields 23888 training patterns in total. We used 90:10 ratio for training and testing sets (five different splits). Parameters for BP were taken from Blank (1997), assuming they were appropriately selected: learning rate 0.1 and momentum 0.7. The overview of results is shown in Table 2. In case of PCA, the learning speed for GHA had to be set to a much lower value 0.0001 (higher values hindered the performance and the value around 0.1 prevented the model from learning completely).

Table 2: Results of of five models combined with implicit and explicit representations.

	implicit		explicit	
Model	error	#ep	error	#ep
AB1	12.60%	400	2.2%	200
AP1	0.60%	75	2.0%	100
AS1	0.40%	75	1.0%	75
AB2	0.40%	50	0.1%	35
AP2	0.14%	35	0.0%	100
AS2	0.19%	45	0.04%	45

We started with AB1-I model, that had at the input $49 \times 6 = 294$ neurons representing the scene, 49 neurons representing the position of the selected object of the source scene and one bias input, together 344 input neurons. The hidden layer consisted of 49 + 1 = 50 neurons and the output layer contained 49 + 49 = 98 neurons. The AB1-I model did not manage to learn both types of analogy even when trained for a large number of epochs ($\sim$1000). Average testing error was 12.6% which roughly corresponds to 310 incorrectly made analogies. Therefore we considered the explicit representation (AB1-E model), in order to test its possible benefit in analogy learning. We used 24 input neurons representing the scene and 49 neurons representing the position of the selected object in the source scene, amounting to 24 + 49 + 1 = 74 input neurons (hence, a much smaller input size). The hidden and output layers were the same as in the previous experiment, i.e. 50 hidden and 98 output neurons. AB1-E model revealed a significant improvement but it still did not manage to simultaneously learn both types of analogy well enough. Testing error remained at about 2.2%. This model also required a smaller number of epochs in order to converge ($\sim$200) and it also had smaller time complexity (thanks to smaller input).

Inclusion of the PCA module (AP1 model) led to a significantly improved performance, in case of implicit representation, both in terms of accuracy (0.6%) and convergence time (75 epochs). Explicit representation did not help in this model, unlike the basic model.

In case of AS1 model, we split the input, so the hidden layer was no longer tied to the second group of inputs. Hence we increased the size of the hidden layer to 100 neurons. Simulations have shown that this model was able to successfully learn both types of analogy with even smaller number of epochs. After 75 epochs the model converged to testing error 0.4%. Since explicit representation had a positive impact on accuracy in the basic model, we tested this effect also at this point, using the AS1-E model. It had identical parameters to the previous model, except for input and output sizes. The model showed slightly worse results and after 75 epochs, testing error was 1.0% (vs. 0.4% in case of AS1-I). This experiment showed that in AS1 model does not benefit from either type of representation.

The last three models were equipped with two hidden layers, both with 100 neurons. In AB2 model we used identical parameter setting as in AB1. Interestingly, both implicit and explicit versions of this model worked very well (accuracies 0.4% and 0.1%), whereas the latter had a little bit faster convergence (50 vs 35 epochs).

As a best model, the combination of two hidden layers with PCA (AP2 models) led to perfect generalization (0%) in case of explicit representation, and quite a small error (0.14%) with implicit representations. The error-free AP2-E model took somewhat longer to converge (100 epochs), though.

Finally, the combination of input splitting with two hidden layers (AS2 models) also led to converging models with both types of representations. Not the best ones (errors 0.19% and 0.04%) but very fast ones (45 epochs).

As a next step, we wanted to get insight into a well trained model. We recorded hidden-layer activations after training, taken from step 1 (which serves as context information for step

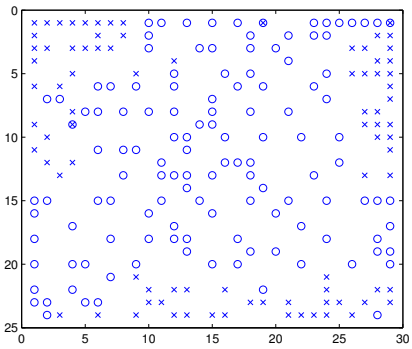

Figure 10: The SOM winners for hidden-layer activation vectors of the trained AS1-I model, corresponding to two types of categorical analogy. Symbol 'x' denotes type 1 (differs in color), and symbol 'o' denotes type 2 (differs in shape).

2). We anticipated that the hidden layer had learned to organize its space by splitting the categories. The best separability could be seen in a AS1 model with 50 hidden neurons, trained on categorical analogies only (differs in color or shape), so we present it here (in order models, the hidden organization was not so well discernible). These patterns (with recorded category labels) were then submitted during training to the self-organizing map (SOM; Kohonen 1982 with 25×29 neurons. For each 50-dim. input pattern, the position of the best matching unit was recorded after training. Figure10 displays the superimposed both classes in the same map. It is evident that the classes are well separated suggesting that the two subsets of hidden vector activations are well separable in the hidden space, hence enabling error-free generalization.

Conclusion

In this article we focused on the problem of geometrical analogies that are a subset of *part–whole* analogy, and we attempted to solve this problem using a connecionist system. The original *Analogator* model (Blank, 1997) was not able to learn both types of analogy (rotational and categorical) so we designed improvements of this model. We also tried using explicit representations, and showed that they had a positive impact in some of the models.

In total, we have evaluated behaviour of the original model and five modifications. The best model combines PCA and two hidden layers, using explicit representation. This AP2-E model learned the task with testing error 0.0% in every simulation. Its minor drawback is that the higher number of training epochs (in comparison to other models) and also a higher time complexity for each iteration. On the other hand, it only needed 8500 pairs for training (around 35%), as opposed to twice as many pairs in case of other models.

We did not thoroughly search for optimal parameters, and used the ones in Blank (1997). The following research in this topic could also investigate our models on different part–whole analogies in order to test their potential.

Acknowledgments

This work was supported by projects no. 1/0898/14 (VEGA) and 076UK-4/2013 (KEGA).

References

Blank, D. (1997). *Learning to See Analogies: A Connectionist Exploration*. PhD thesis, Indiana University, Bloomington.

Chalmers, D. J., French, R. M., and Hofstadter, D. R. (1992). High-level perception, representation, and analogy: A critique of artificial intelligence methodology. *Journal of Experimental and Theoretical Artif. Intelligence*, 4:185–211.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14(2):179–211.

Evans, T. G. (1968). A program for the solution of a class of geometric-analogy intelligence-test questions. In Minsky, M. L., editor, *Semantic Information Processing*, pages 271–353. MIT Press, Cambridge, Massachusetts.

Falkenhainer, B., Forbus, K., and Gentner, D. (1986). The structure-mapping engine. In *Proc. of the 5th National Conf. on Artif. Intelligence*, pages 272–277, Philadelphia, PA.

Forbus, K., Usher, J., Lovett, A., Lockwood, K., and Wetzel, J. (2008). Cogsketch: Open-domain sketch understanding for cognitive science research and for education. In *Eurographics Workshop on Sketch-Based Interfaces and Modeling*. The Eurographics Association.

Gentner, D. and Forbus, K. D. (2011). Computational models of analogy. *Wiley Interdisciplinary Reviews: Cognitive Science*, 2(3):266–276.

Gillan, D., Premack, D., and Woodruff, G. (1981). Reasoning in the chimpanzee: I. analogical reasoning. *Journal of Exper. Psychology: Animal Behavior Processes*, 7(1):1–17.

Halford, S. G., Wilson, W., Guo, J., Gaylor, R. W., Wiles, J., and Stewart, J. E. M. (1994). Connectionist implications for processing capacity limitations in analogies. In *Advances in Connectionist and Neural Computation Theory, Volume 2: Analogical Connections*, pages 363–445. Ablex Publishing, Norwood, New Jersey.

Hall, R. P. (1989). Computational approaches to analogical reasoning: A comparative analysis. *Artif. Intelligence*, 39:39–120.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biol. Cybernetics*, 43:59–69.

Lovett, A., Tomai, E., Forbus, K. D., and Usher, J. M. (2009). Solving geometric analogy problems through two-stage analogical mapping. *Cognitive Science*, 33:1192–1231.

Mitchell, M. and Hofstadter, D. (1995). Perspectives on copycat: Comparisons with recent work. In *Fluid Concepts and Creative Analogies*, pages 275–299. Basic Books, Inc., New York, NY.

Sanger, T. D. (1989). Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural Networks*, 2:459–473.