UC Merced
Proceedings of the Annual Meeting of the Cognitive Science Society

Title
Learning the Structure of Abstract Groups

Permalink
https://escholarship.org/uc/item/0rb9m2jt

Journal
Proceedings of the Annual Meeting of the Cognitive Science Society, 31(31)

ISSN
1069-7977

Authors
Schlimm, Dirk
Shultz, Thomas

Publication Date
2009

Peer reviewed

eScholarship.org Powered by the California Digital Library
University of California

Learning the Structure of Abstract Groups

Dirk Schlimm (dirk.schlimm@mcgill.ca)
Department of Philosophy and School of Computer Science, McGill University
855 Sherbrooke St. W., Montreal, QC H3A 2T7, Canada

Thomas R. Shultz (thomas.shultz@mcgill.ca)
Department of Psychology and School of Computer Science, McGill University
1205 Penfield Avenue, Montreal, QC H3A 1B1, Canada

Abstract

It has recently been shown that neural networks can learn particular mathematical groups, for example, the Klein 4-group (Jamrozik & Shultz, 2007). However, there are groups with any number of elements, all of which are said to instantiate the *abstract group structure*. Learning to differentiate groups from other structures that are not groups is a very difficult task. Contrary to some views, we show that neural networks can learn to recognize finite groups consisting of up to 4 elements. We present this problem as a case study that exhibits the advantages of knowledge-based learning over knowledge-free learning. In addition, we also show the surprising result that the way in which the KBCC algorithm recruits previous knowledge reflects some deep structural properties of the patterns that are learned, namely, the structure of the subgroups of a given group.

Keywords: Mathematical groups; neural networks; KBCC; CC.

Introduction

Mathematical groups are remarkable for two reasons: On the one hand, they pervade many areas of mathematics and also everyday life, and on the other hand, their abstract structure is quite difficult to learn from instances. In this paper we use the task of learning the abstract structure of small finite groups as a case study to make the following three points: (a) We provide an argument against purported limitations of neural network approaches. (b) We show how neural networks can learn from previously existing knowledge and that this increases the speed of their learning. (c) Our results reveal that the neural networks show some 'deep understanding' of the abstract structure of the problems they are learning.

Some authors have argued that ordinary artificial neural networks are inherently unable to learn systematically structured representations such as mathematical groups (Phillips, & Halford, 1997; Marcus, 1998). However, a first step has recently been made to invalidate this claim, because it has been demonstrated that neural networks can learn particular mathematical groups, like the Klein 4-group, in a fashion that simulates learning by humans (Jamrozik & Shultz, 2007). We now extend the task to learning the abstract structure of small finite groups with three and four elements and we compare knowledge-based learning with knowledge-free learning.

Artificial neural networks most always learn from scratch, unlike people who almost always try to build new learning on top of their existing knowledge. Our results show how neural networks can also learn from existing knowledge. In particular, we are curious to know whether it helps the learning process to have previously learned the structure of smaller groups. This is especially interesting, because similarities between two different groups and differences between groups and other domains cannot in general be characterized by structure-preserving mappings, but only by formulating their underlying laws (Schlimm, 2008).

Learning of group structure based on previous knowledge of smaller groups is a natural area of application for constructive algorithms that build their learning on top of existing knowledge. One such algorithm is knowledge-based cascade-correlation (KBCC), which constructs a neural-network topology by recruiting previously learned networks and single hidden units (Shultz & Rivest, 2001). Its performance can be readily compared to ordinary cascade-correlation (CC), which builds a network only by recruiting single hidden units (Fahlman & Lebiere, 1990). Past comparisons have revealed that KBCC tends to recruit relevant knowledge whenever it can and that this speeds learning (Shultz & Rivest, 2001) and in some cases makes learning possible (Egri & Shultz, 2006).

Finally, we investigate the relation between the group to be learned and the kinds of networks that are recruited from the knowledge base: Does only the size of the recruits matter, or does the quality of their knowledge also play a role? Our results provide a surprising answer to this question that points to promising new lines of work on knowledge and learning.

The Abstract Group Structure

A mathematical group consists of a set of elements together with an associative operation * (i.e., where $a*(b*c)$ and $(a*b)*c$ yield the same result), for which the following three properties hold: (i) the objects are closed under the group operation, i.e., for elements a and b, $a*b$ is also an element of the group; (ii) there exists a neutral element n, such that $a*n=a$ for all elements a of the group; and (iii) that for every element a there exists an inverse element a', such that $a*a'=n$.

Typical examples of groups are the first n-1 natural numbers with addition modulo n as the group operation (for n=3, see Figure 1; for n=4, see the cyclic 4-group in Figure 3); the positions of a cube with the operation of performing rotations in space; all possible permutations of a given sequence of objects with successive application of the permutation as the group operation (Rothman, 1985); and, the different configurations that can be obtained by flipping a mattress (Hayes, 2005). The group operation is usually presented by a multiplication table, or Cayley table, as in the examples shown in Figure 1.

*	**0**	**1**	**2**
0	0	1	2
1	1	2	0
2	2	0	1

*	**b**	**a**	**c**
b	a	c	b
a	c	b	a
c	b	a	c

Figure 1: Addition modulo 3 and alternative presentation.

The multiplication table on the left in Figure 1 represents a 3-element group with elements 0, 1, 2, where the group operation * is interpreted as addition modulo 3; the neutral element is 0; as one can see by inspecting the table, the inverse of 0 is 0, the inverse of 1 is 2, and the inverse of 2 is 1. Because an abstract group is characterized *only* by the structure induced by the group operation and not by the names of the elements or the order in which they are presented in a multiplication table, renaming the elements and/or rearranging the rows and columns yields a representation of the *same* abstract group (see, for example, the multiplication table on the right in Figure 1 for an alternative presentation; here c is the neutral element). In fact, one can show that there is *only one* abstract group with three elements (Rotman, 1996). In other words, any set with three elements and an operation that satisfies the conditions (i)–(iii) for groups listed above can be put into the form of the multiplication tables in Figure 1 by renaming the elements and rearranging the rows and columns.

There is also exactly one abstract group with two elements, which is represented by the two multiplication tables in Figure 2 (the group on the left has the neutral element 0, while the group on the right has the neutral element 1; these two groups are again instances of the same abstract group).

*	**0**	**1**
0	0	1
1	1	0

*	**0**	**1**
0	1	0
1	0	1

Figure 2: Two alternative representations of the same abstract 2-element group.

There are two *different* abstract groups with four elements: the cyclic 4-group, and the Klein 4-group (see Figure 3). They are considered different because the multiplication table of a Klein 4-group can never be made to coincide with that of a cyclic 4-group by renaming the elements and rearranging the rows and columns. In mathematical terms, these two groups are not *isomorphic*.

*	**0**	**1**	**2**	**3**
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

Cyclic 4-group

*	**0**	**1**	**2**	**3**
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

Klein 4-group

Figure 3: Two different 4-element groups, the cyclic 4-group (left) and the Klein 4-group (right).

Both 4-element groups have the 2-element group as a *subgroup*. In other words, within their 4x4 multiplication tables one can identify pairs of elements that behave like a group with 2 elements. For the cyclic 4-group these are the elements 0 and 2, while the Klein 4-group has three such pairs: elements 0 and 1, 0 and 2, and 0 and 3. Notice also, that the 3-element group (Figure 1) does not have the 2-element group as a subgroup. This difference between 3- and 4-element groups has interesting implications for interpreting the results of knowledge-based learning of groups. We return to this difference in the discussion of our results.

For our research, we trained neural networks with all possible multiplication tables for the 3-element and the 4-element groups using knowledge-free (CC) and knowledge-based (KBCC) learning algorithms in order to assess their abilities, and, in the case of the knowledge-based algorithm, also to study the quality of the recruited knowledge.

Method

CC Learning

CC learning alternates between two phases: input phase and output phase (Fahlman & Lebiere, 1990). Each phase is composed a number of epochs, where an epoch is a pass through the set of training examples. CC networks begin learning without any hidden units. They start training in output phase, by adjusting connection weights entering output units to reduce as much error as possible. In input phases, the input weights to candidate hidden units are trained in order to maximize the covariance between unit activation and network error. The candidate unit with the highest absolute covariance is selected and installed into the network with random input connection weights of the same sign as those that were just learned, the other candidates are discarded, and training shifts back to output phase. The algorithm shifts from one phase to the other when the current phase fails to improve the solution of the problem on which the network is being trained. That is, it fails to reduce error in output phase, or fails to increase covariances in input phase. CC and related algorithms have been used to simulate many aspects of cognition and its development (Shultz, 2003; 2006).

KBCC Learning

KBCC differs from CC mainly by having the potential to recruit previously-learned networks in competition with single hidden units (Shultz & Rivest, 2001). The computational device that gets recruited is the one whose output covaries best with residual network error. An example of a KBCC network is shown in Figure 4, illustrating that a recruited source network can have multiple inputs and outputs, thus requiring connection-weight matrices rather than vectors. Mathematical details for KBCC are available elsewhere (Shultz & Rivest, 2001), along with evidence of its use in cognitive simulations (Shultz, Rivest, Egri, Thivierge, & Dandurand, 2007).

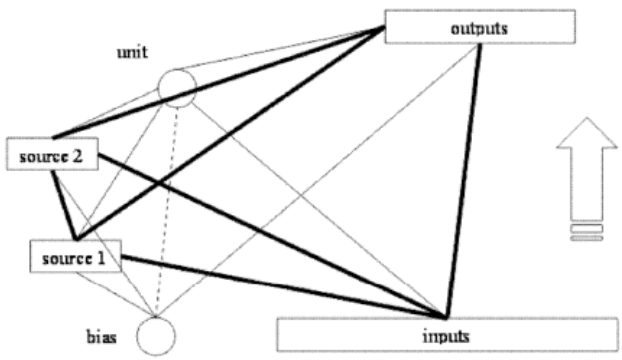

Figure 4: Drawing of sample KBCC network after recruiting two source networks and one sigmoid unit. The dashed line represents a single connection weight, the light solid lines represent weight vectors, and the dark solid lines represent weight matrices. Activation flow is upward.

The 3x3 Learning Task

As inputs we encode the group multiplication tables discussed earlier, in which the names and the order of the group elements are kept fixed. In this case the 3-element group can be presented by three different multiplication tables, as shown in Figure 5. (In the first multiplication table 0 is the neutral element, while it is 1 and 2 in the second and third table, respectively). Because these tables can be transformed into each other by renaming the elements and rearranging the rows and columns, they all represent the same abstract group. Also, because the names of the elements and their order are kept constant, we can omit the labels of the columns and rows from the inputs, so that the multiplication table of a 3-element group can be given as a 3x3 matrix.

*	**0**	**1**	**2**
0	0	1	2
1	1	2	0
2	2	0	1

*	**0**	**1**	**2**
0	2	0	1
1	0	1	2
2	1	2	0

*	**0**	**1**	**2**
0	1	2	0
1	2	0	1
2	0	1	2

Figure 5: Alternative presentations of the 3-element group.

When presenting a group multiplication table it is customary to always write the neutral element in the leftmost column and the top row, as in the table on the left in Figure 5. However, we did not employ this convention for the inputs in order to prevent networks from learning some feature that is purely an artifact of this particular mode of presentation, e.g., that 0*0 is always 0. As a consequence, the learning task itself becomes much more difficult. To grasp this, consider the task of determining whether the two multiplication tables presented in Figure 1 or the three tables shown in Figure 5 are indeed instances of the same abstract group. Even most trained mathematicians would not be able to solve this at a glance.

It is noteworthy that only very few multiplication tables actually represent instances of an abstract group. A 3x3 multiplication table, or matrix, can be filled with at most three elements in 3^9=19,683 different ways. If only those tables are considered in which each element occurs exactly three times, this reduces the number of different multiplication tables to 1,680. If the tables are further restricted to only those in which each element occurs exactly once in each row and column, 12 different tables remain, of which only the three shown in Figure 5 represent the abstract group structure.

The candidate units for the KBCC algorithm consisted of multiple instances of a sigmoid unit, a network that was trained to learn the 2x2 group using the CC algorithm (genuine 2x2), and a network of the same structure as the previous one, but with random weights chosen with uniform distribution from the range -1 to 1 (random 2x2). As training data for the genuine 2x2, we used all 16 possible configurations of the elements 0 and 1 in a 2x2 multiplication table, only two of which are in fact groups (these two are shown in Figure 3).

For the comparison of the CC and the KBCC algorithms we used in the training set 900 positive matrices, i.e., 300 copies of the three alternative presentations of the 3-element group, and 1000 random 3x3 matrices, in which every element occurs exactly three times, that are not groups.

The 4x4 Learning Task

Each of the two abstract groups with four elements can be represented by 16 different 4x4 matrices (if the names of the elements and their order are kept fixed). Of these, 15 were used in the training sets and one was used for testing for generalization. As in the 3x3 case, we again used 900 positive instances, i.e., 60 copies of the 15 4x4 group multiplication tables and 1000 random 4x4 matrices that are not groups in which every element occurs exactly four times.

The pool of possible recruits by the KBCC algorithm consisted of multiple instances of the same units as in the 3x3 learning task (sigmoid, genuine 2x2, and random 2x2), plus two additional 3x3 networks. The genuine 3x3 network had learned the 3-element group structure using a training set consisting of 1090 matrices in total, 90 of which were copies of the 3 different multiplication tables for the 3x3 group. The random 3x3 network had the same structure as

the genuine 3x3, but again with random weights chosen with uniform distribution from the range -1 to 1.

Both the 3x3 and the 4x4 learning tasks were run 20 times to enable statistical comparisons. Each network was trained until the single output unit activation was within the score-threshold parameter of its target value on every training pattern. Because the output decision was a binary one, i.e., the input matrix is a group or is not a group, the score-threshold was the default value of 0.4. The sigmoid activation function has an output in the range -.5 to .5.

Results

We analyzed learning speed, number of recruits, the quality of KBCC recruits, and generalization ability.

For learning speed, we performed a task dimensions x algorithm factorial ANOVA of the epochs required to learn the training patterns. There were main effects of dimensions $F(1, 76) = 7.37$, $p < .01$ and algorithm $F(1, 76) = 15.54$, $p < .001$, but no significant interaction. The means are presented in Figure 6. The 4x4 patterns took longer to learn than the 3x3 patterns, and CC took longer to learn than did KBCC. The speedup provided by KBCC was significant for both 3x3 patterns, $t(38) = 3.51$, $p < .001$, and 4x4 patterns $t(38) = 2.31$, $p < .03$. Thus, KBCC learned these groups faster than did knowledge-free CC.

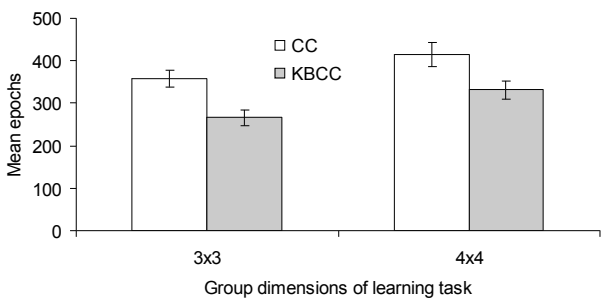

Figure 6: Mean epochs to learn groups of different dimensions by two algorithms ±SE.

Number of recruits was also subjected to a dimensions x algorithm factorial ANOVA. There were two main effects and no interaction. The main effect of dimensions reflected more recruits for the 4x4 task ($M = 3.98$) than for the 3x3 task ($M = 3.2$), $F(1, 76) = 5.34$, $p < .03$. The main effect of algorithm revealed that CC used more recruits ($M = 4.00$) than did KBCC ($M = 3.17$), $F(1, 76) = 6.05$, $p < .02$. Generally in CC-type algorithms, the more recruits that are required, the longer it takes to learn.

For quality of recruits, we analyzed the 3x3 task separately from the 4x4 task because some types of recruits were available in the latter but not in former. Learning 3x3 patterns, KBCC networks recruited many more genuine 2x2 source networks ($M = 2.05$) than randomized 2x2 source networks ($M = 0.35$), $t(19) = -6.47$, $p < .001$. This is the pattern one would expect if the recruits' actual knowledge is more important than its mere complexity. See Table 1 for the proportions of the units recruited by KBCC in both learning tasks.

Table 1: Proportions of recruits by KBCC.

	Dimensions of group to be learned	
Recruits	3x3	4x4
Sigmoid	.077	.147
Genuine 2x2	.788	.427
Random 2x2	.135	.067
Genuine 3x3	n/a	.227
Random 3x3	n/a	.133

For learning 4x4 patterns, the counts of genuine and random 2x2 and 3x3 recruits were subjected to a type of recruit x dimensions of recruit repeated measures ANOVA. The associated means are plotted in Figure 7. There was a main effect of type of recruit and an interaction between that and the dimensions of the recruit. The main effect of type of recruit, $F(1, 19) = 10.34$, $p < .005$, reflects that recruits were more often genuine than random, again the expected pattern if knowledge trumps mere complexity. The interaction, $F(1, 19) = 8.64$, $p < .01$, was further explored with dependent t tests. Among genuine network recruits, there were more 2x2s than 3x3s, $t(19) = 3.00$, $p < .01$, an interesting finding discussed more fully in the Discussion section. Whereas among random network recruits, there was no significant difference related to group dimensions of the recruit, $t(19) = -1.42$, *ns*.

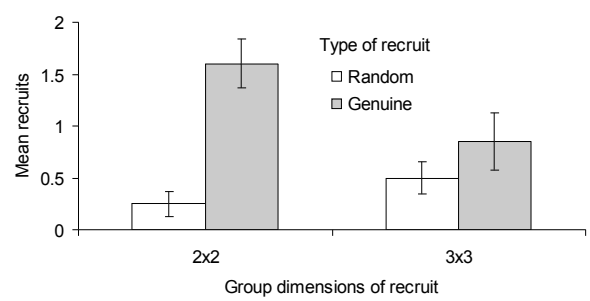

Figure 7: Mean genuine and random network recruits of different group dimensions in KBCC learning of 4x4 groups ±SE.

For the 4x4 learning task we tested generalization in 20 fresh CC and 20 fresh KBCC networks by recording each network's error on 1 of the 16 sets of patterns that was not used in training. The mean errors are plotted in Figure 8. A repeated-measures ANOVA on these errors with algorithm as a between network factor and training as a within network factor yielded only a main effect of training, $F(1, 38) = 71.42$, $p < .0001$. Error dropped sharply on these test patterns during training with either algorithm.

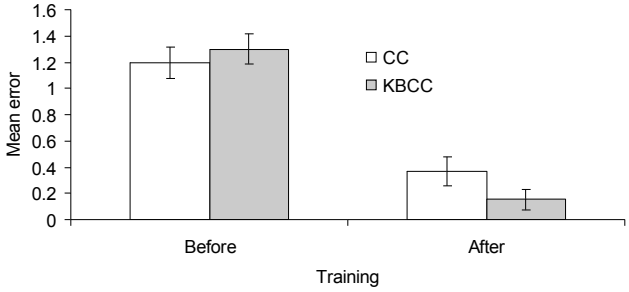

Figure 8: Mean error on test patterns before and after training ±SE.

Discussion

It is evident that both CC and KBCC algorithms are able to learn the structure of the abstract 3-element and 4-element groups, and that they generalize successfully to untrained test patterns. Thus, claims that such tasks are impossible to learn by neural networks are successfully refuted by our results.

Comparison between the CC and KBCC learning algorithms reveals the following: (a) KBCC consistently learns faster than CC, and (b) KBCC consistently has fewer recruits. These two differences are due to the fact that KBCC chooses recruits that possess knowledge that is relevant for the task at hand. Thus, our results confirm similar learning speedups due to knowledge that were obtained previously with very different learning tasks (Shultz & Rivest, 2001).

In addition, if we look at the types of networks that are recruited by KBCC, we notice very distinctive preferences (see Table 1), which cannot be explained in terms of the complexities of the recruited networks alone. When learning the 3x3 group, KBCC recruited in the majority of cases the genuine source network, i.e., the network that had already learned the 2x2 group. This shouldn't come as too big a surprise, because both groups share those structural features that are characteristic of all groups and that distinguish them from other patterns that do not instantiate the abstract group structure. (These features are the properties (i), (ii), and (iii), presented earlier.) This also explains why, in the 4x4 learning task, KBCC recruited the two networks that were trained with the smaller groups (Genuine 2x2 and Genuine 3x3) more frequently than the networks with random weights or single sigmoid units.

But, why did KBCC recruit the genuine 2x2 network significantly more often than the genuine 3x3 network? After all, because the 3x3 network is larger it should be able to encode more information than the 2x2 network. Some insight into the structural relations between the different 2, 3, and 4-element groups can resolve this mystery and explain the prima facie irrational behavior of the KBCC algorithm.

As mentioned earlier, both of the abstract 4x4 groups have 2x2 groups as their *subgroups*. Thus, the 2x2 group is a sort of building block for both the cyclic 4-group and the Klein 4-group, and the KBCC algorithm picks up on these subtle structural relationships. In other words, in addition to being sensitive to the abstract group structure, KBCC is also able to exploit deeper structural relationships between what is being learned and what is already present in the knowledge base. In the present case this has the effect that those candidate networks are preferred that represent subgroups of the group that is being learned. To summarize, our results suggest that KBCC does not recruit networks primarily based on some quantitative measure (e.g., size and complexity), but based on the relevance and quality of the knowledge they encode.

In future work we would like to investigate if this observation can also be substantiated when larger groups are being learned. Based on our experiences so far we expect that in these cases further structural properties, like that of being a cyclic group or being a commutative group, will play a role for the selection of the candidate recruits.

As anticipated, the learning of the abstract group structure proved to be a worthy and interesting challenge for constructive network algorithms. CC is known to be a strong learner because of the way it searches simultaneously in both topology space and weight space to find, not only appropriate connection weights, but also the right topology for the task it is learning (Fahlman & Lebiere, 1990; Shultz, 2003). Thus, CC naturally, without intervention of programmers, builds networks of about the right size for the learning task. Consequently, it can learn to distinguish groups from non-groups, given enough learning time and examples. But KBCC, by recruiting simpler relevant knowledge than its training task, can learn these distinctions even faster and with fewer recruits than CC can. There is little question that these constructive neural learners can, from examples alone, learn important features about abstract, systematic structures.

We make no claim that these neural networks understand groups in the same explicit fashion as mathematical specialists do. But these algorithms could well serve to model how ordinary people acquire complex, abstract, highly-structured knowledge (Jamrozik & Shultz, 2007; Egri & Shultz, 2006). Modeling expert explicit mathematical knowledge would require in addition that this implicit learned knowledge be converted into an axiomatic characterization that expresses the three group properties (i)–(iii) listed in our earlier section on The Abstract Group Structure, to allow for a concise representation of both finite and infinite groups (Schlimm, 2008).

Acknowledgements

This research is supported by a grant to DS from the Social Sciences and Humanities Research Council of Canada, and a grant to TRS from the Natural Sciences and Engineering Research Council of Canada. We also would like to thank James Olchowski for helping to run the learning tasks.

References

Egri, L., & Shultz, T. R. (2006). A compositional neural-network solution to prime-number testing. In R. Sun & N. Miyake (Eds.), *Proceedings of the Twenty-eighth Annual Conference of the Cognitive Science Society* (pp. 1263–1268). Mahwah, NJ: Lawrence Erlbaum.

Fahlman, S. E., & Lebiere, C. (1990). The cascade-correlation learning architecture. In D. S. Touretzky (Ed.), *Advances in neural information processing systems 2* (pp. 524–532). Los Altos, CA: Morgan Kaufmann.

Hayes, B. (2005) Group theory in the bedroom. *American Scientist*, 93, 395–399.

Jamrozik, A., & Shultz, T. R. (2007). Learning the structure of a mathematical group. In D. McNamara & G. Trafton (Eds.), *Proceedings of the Twenty-ninth Annual Conference of the Cognitive Science Society* (pp. 1115–1120). Mahwah, NJ: Lawrence Erlbaum.

Marcus, G. F. (1998). Rethinking eliminative connectionism. *Cognitive Psychology, 37*, 243–282.

Phillips, S., & Halford, G. S. (1997). Systematicity: Psychological evidence with connectionist implications. In M. G. Shafto & P. Langley (Eds.), *Proceedings of the Nineteenth Annual Conference of the Cognitive Science Society*. Mahwah, NJ: Erlbaum.

Rothman, T. (1982). The short life of Évariste Galois. *Scientific American*, *246* (4), 136–149.

Rotman, J. J. (1996). *A first course in abstract algebra*. Upper Saddle River, NJ: Prentice Hall.

Schlimm, D. (2008). Two ways of analogy: Extending the study of analogies to mathematical domains. *Philosophy of Science*, 75 (2), 178–200.

Shultz, T. R. (2003). *Computational developmental psychology*. Cambridge, MA: MIT Press.

Shultz, T. R. (2006). Constructive learning in the modeling of psychological development. In Y. Munakata & M. H. Johnson (Eds.), *Processes of change in brain and cognitive development: Attention and performance XXI*. (pp. 61–86). Oxford, UK: Oxford University Press.

Shultz, T. R., & Rivest, F. (2001). Knowledge-based cascade-correlation: Using knowledge to speed learning. *Connection Science, 13*, 1–30.

Shultz, T. R., Rivest, F., Egri, L., Thivierge, J.-P., & Dandurand, F. (2007). Could knowledge-based neural learning be useful in developmental robotics? The case of KBCC. *International Journal of Humanoid Robotics, 4*, 245–279.