Modeling Acquisition of a Torque Rule on the Balance-scale Task

Frederic Dandurand (frederic.dandurand@univ-provence.fr)
Laboratoire de psychologie cognitive, CNRS & Université de Provence, UMR 6146, Case D, Bâtiment 9,
3 Place Victor Hugo, 13331 Marseille Cedex 3 France

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

We present a new model of development of children's performance on the balance-scale task, one of the most common benchmarks for computational modeling of development. Knowledge-based cascade-correlation (KBCC) networks progress through all four stages seen in children, ending with a genuine torque rule that can solve problems only solvable by comparing torques. A key element in the model is injection of a neurally-implemented torque rule into the recruitment pool of KBCC networks, mimicking the explicit teaching of torque in secondary-school science classrooms.

Keywords: Cognitive development; balance scale; neural networks; knowledge-based learning; KBCC; SDCC.

Introduction

The ongoing competition between symbolic and neural-network models of cognition often focuses on development of children's performance on balance-scale problems, one of the most modeled tasks in developmental psychology. The symbolic view is that knowledge is represented in rules containing propositions referring to things in the world, that processing occurs as rules are selected and fired thus generating new propositions, and that knowledge is acquired by learning such rules. In neural-network accounts, active knowledge is represented in rapidly changing unit activations and long-term knowledge by excitatory and inhibitory connections between units, processing involves activations being passed from one layer of units to another, and knowledge acquisition results from adjustment of connection weights and occasional recruitment of new units into the network. The symbolic approach is sometimes referred to as *rule use*, and the neural-network approach as *rule following* (Shultz & Takane, 2007).

At first glance this may seem to be a rather subtle distinction, but there are important differences between the two viewpoints that have consistently guided research over the last few decades. The rule-use approach assumes that people literally have and use rules to guide their reasoning and behavior, often affording the perfect generalization that symbolic rules sometimes allow. Rule-use is consistent with the idea that human cognition is often quite regular. In contrast, the rule-following approach assumes that such regularities may be approximated by neural networks that adapt to regularities in the environment. This affords graded generalizations whose regularity approximates the extent to which the environment is consistently regular, with the advantage that both regularities and exceptions, which are quite common in the complex phenomena that humans encounter, can be handled within the same neural network. In rule-use approaches, exceptions are instead typically memorized by a separate system from the rules themselves. Such differences are highlighted as researchers build precise computational models of psychological theories (Shultz, 2003). Computational models with artificial neural networks are quite different from those that represent and use symbolic rules.

One of the most frequently modeled domains in developmental psychology focuses on the balance-scale task, used by Siegler (1976) and several other developmental researchers. The balance-scale is considered to be representative of the many tasks requiring integration of information across two separate quantitative dimensions. Results have been consistently replicated and include an interesting stage progression.

Here we report an extended computational model of balance-scale development that addresses a recent criticism affecting most of the computational models – namely ensuring that the final stage consists of a genuine, multiplicative torque rule and not a simpler rule based on addition (Quinlan, van der Maas, Jansen, Booij, & Rendell, 2007). We first describe the basic balance-scale task and its stages before presenting our new computational model.

The Balance-scale Task

In this task, a participant is presented with a rigid beam balanced on a fulcrum (Siegler, 1976). There are several pegs positioned on the beam at regular distances to the left and right of the fulcrum. An experimenter places some identical weights on a peg on the left side and some other identical weights on a peg on the right side of the scale. The participant is asked which side of the scale will drop, or whether the scale will remain balanced, when the beam is released from its supports, often consisting of a block placed under each end of the beam. Archimedes' (c. 287-212 BC) principle of the lever describes a rule that yields a correct answer to balance-scale problems: multiply the weight and distance from the fulcrum on each side and predict the side with the larger product (torque) to drop.

A neural-network simulation using the cascade-correlation (CC) algorithm (Shultz, Mareschal, & Schmidt, 1994) captured the four stages seen in children (Siegler,

1976): 1) predicting the side with more weights to descend; 2) when the weights are equal on both sides, also predicting the side with greater distance to descend; 3) predicting correctly when weight and distance cues both forecast the same result and performing at chance when these cues conflict, as in the problems shown in Figure 1; and 4) being correct on at least 80% of balance-scale problems.

Diagnosing Stage 4

If performance at Stage 4 is diagnosed as being correct on 80% of balance-scale problems, many of which are difficult problems in which weight and distance cues conflict with each other, then at least some computational models, both symbolic (Schmidt & Ling, 1996) and connectionist cascade-correlation networks (Shultz et al., 1994), succeed in reaching Stage 4. But if Stage 4 is defined by possession of a genuine torque rule, as opposed to a mere addition rule, then the modeling challenge is still open. Because many conflict problems can be solved by adding (rather than multiplying) weight and distance, documentation of a torque rule needs to be supported by success on problems that cannot also be solved by an addition rule (Boom, Hoijtink, & Kunnen, 2001; Quinlan et al., 2007).

With five pegs and five weights, the problem size often used in simulations of the balance scale (Shultz et al., 1994), there are 625 total problems, of which just 200 are relatively difficult conflict problems in which weight and distance information, used alone, lead to different answers. Only 52 of these conflict problems are torque problems that cannot be solved by mere addition; the other 148 conflict problems are addition problems that can be solved correctly by adding distance and weight on each side and comparing these sums.

The torque problems, an example of which is shown in Figure 1a, require comparison of left and right torques. In example 1a, a torque of 6 on the left side is greater than a torque of 4 on the right side. Comparing sums instead predicts this scale will balance because the sum of weight and distance on each side is 5. Most conflict balance-scale problems, such as the one in Figure 1b, can be solved by a simpler addition rule: predict that the side with the larger sum of weight plus distance values will descend. In 1b, the sum on the right side (6) is greater than the sum on the left side (5). Likewise, the torque on the right side (8) is greater than the torque on the left side (6).

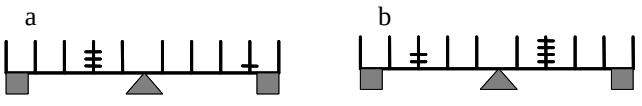

Figure 1: Example of a torque problem (a) that can be solved by comparing torques but not by comparing sums, and an example of an addition problem (b) that can be solved by comparing either sums or torques.

Until recently, addition was routinely ignored in computational models of balance-scale development, whether symbolic (Schmidt & Ling, 1996) or connectionist (McClelland, 1989; Shultz et al., 1994), just as it had been ignored in many older psychology experiments. But with recent evidence that at least some people use or follow a genuine torque rule, solving balance-scale problems that addition cannot solve (Boom et al., 2001; Quinlan et al., 2007), it is important to test computational models on their ability to acquire a genuine torque rule. Some researchers (Quinlan et al., 2007) argued that neural-network models may not be able to learn a genuine torque rule.

However, we showed that constructive neural networks could learn a torque rule in either of two ways: by prolonged training with sufficient numbers of torque problems, or by being taught an explicit torque rule as often happens with adolescents in secondary-school science courses (Shultz, Rivest, Egri, Thivierge, & Dandurand, 2007). The former method was implemented in ordinary CC networks that recruit single hidden units having a sigmoid activation function; the latter with KBCC, permitting recruitment of previously learned networks or injected functions as well as single hidden units (Shultz & Rivest, 2001; Thivierge, Dandurand, & Shultz, 2004).

Our experience teaching university students about psychological development on the balance scale suggests that those few students who spontaneously use the torque rule to solve balance problems admit that they learned this method in science classes, either in secondary school or college. When the remaining students are informed that balance-scale problems can be solved by computing and comparing torques, they too begin to sometimes use this torque rule to produce more correct answers. Thus, it seems likely that most people learn a torque rule from explicit verbal instruction that includes relevant examples (Siegler, personal communication). In contrast, people are unlikely to learn a torque rule from processing many examples alone because problems requiring the torque rule (like that in Figure 1a) are so rare.

Consistent with this idea, we found that knowledge-based learning with KBCC performs better, particularly in making the transition to the correct responding characteristic of a stage-4 torque rule, than networks that learn solely from examples (Shultz et al., 2007). In a variant of KBCC, called function-based CC (FBCC), symbolic functions can be injected into the recruitment pool. The injected function in our recent simulations was a torque-difference function inputting continuous values representing a left and a right weight-and-distance pair, and producing the difference between the left and right torque products that was then squashed through a sigmoid output unit. KBCC can equally well recruit functions or networks, the only restriction being that the recruit is as a differentiable function. These KBCC networks made a transition between stage 3 and stage 4, diagnosed by either the 80%-correct method (Siegler, 1976) or by latent class analysis (Quinlan et al., 2007). However, this model did not capture the progression through the first three stages of balance-scale development.

Here, we attempt to achieve a successful and psychologically more valid model of balance-scale

development by capturing all four stages, including a genuine torque rule at stage 4. The new model combines and extends our initial balance-scale simulation (Shultz et al., 1994) and our recent exploratory work with KBCC (Shultz et al., 2007).

Method

Learning Algorithms

Ordinary CC learns by alternating between two phases: input phase and output phase (Fahlman & Lebiere, 1990). CC networks at first have no hidden units. They begin training in output phase, by adjusting connection weights entering output units to reduce error as much as possible. In input phases, the inputs to candidate hidden units are trained so as 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 just learned, the other candidates are discarded, and there is a shift 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, by not reducing error or failing to improve covariances, for output- or input-phase, respectively.

Sibling-descendant cascade-correlation (SDCC) is a newer version of CC that decides whether to install a new hidden-unit recruit on the current highest layer (as a sibling) or on its own highest layer (as a descendant) (Baluja & Fahlman, 1994). SDCC decides to do whichever is better at the time of installation, recruiting the unit whose activations covary best with existing network error. In each input phase, the candidate pool contains equal numbers of sibling units and descendant units, each with randomly-initialized input weights from the network's input units. Because of the tendency to recruit units with the most computational power, the correlations with descendant candidates (having extra, cascaded inputs from hidden units at the current highest level) are typically penalized by a multiplier of 0.8. This has been found to decrease network depth without damaging generalization to untrained test patterns (Baluja & Fahlman, 1994). When used in a variety of psychology simulations, SDCC performs with the same functionality as standard CC, but with flatter networks, fewer connection weights, and more topological variety (Shultz, 2006).

KBCC differs from CC and SDCC mainly in that it has the potential to recruit previously-learned networks or indeed any differentiable function, 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. A simple example of a KBCC network is shown in Figure 2, illustrating that a recruited source network or function can have multiple inputs and outputs, thus requiring connection-weight matrices rather than vectors. Mathematical details about KBCC are available elsewhere (Shultz & Rivest, 2001; Shultz et al., 2007).

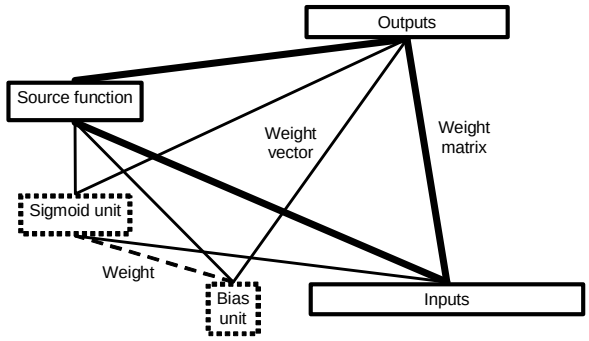

Figure 2: Drawing of a sample KBCC network that has recruited a single sigmoid hidden unit followed by a source function. Thick solid lines represent connection-weight matrices, thin solid lines represent connection-weight vectors, and the dashed line represents a single connection weight.

Torque-rule Injection

To simulate the teaching of a torque rule, we introduce after 350 epochs a module consisting of a KBCC network which has a four-input function (hereafter referred to as the torque rule) in its recruitment source pool:

$$TR = \frac{1}{1+e^{-4TD}} - 0.5 \qquad \text{Equation 1}$$

where $TD = (w_r d_r) - (w_l d_l)$ Equation 2

Here, *TR* is the torque rule, and *TD* is torque-difference, computed as the difference between the torque on the right side of the fulcrum and the torque on the left side of the fulcrum. On each side of the fulcrum, right or left, torque is computed as the product of weight (*w*) and distance (*d*). *TD* is then passed through a sigmoid squashing function to obtain *TR*. *TR* is an S-shaped activation function with a floor at -0.5, a ceiling at 0.5, and an inflection point at 0. *TR* is also a differentiable function, which KBCC requires of potential recruits. The exponent of 4 increases the steepness of *TR*, emphasizing the binary judgments that humans are asked to make on this task, but the reported results were also produced with an exponent of 1.

Simulation Modules

There are four key modules: an intuitive network, a torque-rule network, a confidence network, and a selection module.

Intuitive Network As in our initial simulation (Shultz et al., 1994), the intuitive SDCC network learns to predict balance scale results from learning with examples only. It has four inputs representing distance on the right, distance on the left, weight on the right, and weight on the left. There are two outputs, whose target patterns are coded as follows: 0 0 for balance, +0.5 -0.5 for left heavier, and -0.5 +0.5 for right heavier. The recruitment pool contains eight sibling and eight descendent sigmoid units.

Training begins with 100 initial patterns, randomly selected from the 625 possible balance-scale problems allowed by five weights and five distances from the

fulcrum. In the selection process, there is a .9 bias toward equal-distance problems (in which the weights are placed equally distant from the fulcrum), designed to encourage early use of the weight rule (the side with more weights should descend) under the assumption that children have rather few experiences with physical devices that systematically vary distance from a fulcrum (McClelland, 1989). One new pattern is added in each output epoch, under this same .9 bias. Because items are selected with replacement, random selection of duplicate patterns is permitted.

Exploratory simulations indicate that networks are well into stage 3 by about 350 epochs (see confirming evidence in Results section). Thus, when a network reaches 350 epochs, we allow it to complete the current output phase, and then stop training.

Parameter settings are the same as in the original simulation: score threshold = 0.25, output learning rate = 0.175, input learning rate = 0.5, and other parameters are left at default values.

Torque-rule Network The torque-rule network uses KBCC with a torque rule as injected knowledge. The target torque-rule network has the same inputs and outputs as the intuitive network. But in the recruitment pool, it has eight sibling torque rules and eight descendent torque rules in addition to the eight sibling sigmoid and eight descendent sigmoid units. Its training data uses the final data set of an intuitive network and expands it with 25 randomly-selected torque problems. There is no further per-epoch expansion of the training set. Parameter settings are the same as for the intuitive network. Training begins in input phase rather than the usual output phase.

Confidence Network The confidence network learns to predict the accuracy of the intuitive network. It has seven inputs. In addition to the usual four inputs describing a balance-scale problem, these include a torque-difference measure, the absolute value of Equation 2, and two binary inputs indicating symmetry of weights and symmetry of distance (1 if symmetrical, 0 otherwise). There is one output encoding (2 – error) / 2, where error is the sum of the absolute values of intuitive network error across the two outputs of the intuitive network on the same problem. This output can be considered as a measure of confidence in the correctness of the intuitive network's answers: as error diminishes toward 0, confidence approaches 1; and as error increases, confidence decreases toward 0. The training data and parameter settings are the same as for the torque-rule network, except that score-threshold is lowered to 0.1, to more accurately learn this continuous confidence function.

Selection Module The selection module is software that decides whether to use the intuitive network or the torque-rule network to generate a response. It first presents the problem to the intuitive and confidence networks, and reads the output of the latter as confidence in the intuitive network. If this confidence is sufficiently high (0.95 or more), it returns the intuitive response, otherwise it returns the torque-rule network's response.

Test Sets

The system is tested with three different sets of problems, labeled Siegler-TD, Addition, and Torque.

Siegler-TD The so-called Siegler-TD test set contains 24 balance-scale patterns selected as in our original simulation (Shultz et al., 1994), inspired by Siegler's (1976) test set but additionally balanced for torque-difference effects. It contains four randomly-selected problems of each of Siegler's six types: balance, weight, distance, conflict balance, conflict weight, and conflict distance problems. The four problems of each of the six problem types each represent a different level of torque difference: 1, 3-5, 6-9, or 10-19. This is an improvement over studies that ignore torque differences and thus risk confounding problem type with torque difference and studies that use only small torque differences and thus risk underestimating torque-difference effects (Shultz et al., 1994).

This test set is used to diagnose stages 0-4 according to Siegler's (1976) criteria, with the proviso that Stage 2 is given diagnostic priority over Stage 3 (Shultz et al., 1994). Rule diagnosis is conducted by software: diagnosis of Stage 4 requires 20 of 24 problems correct; diagnosis of stage 2 requires at least 13 correct on the 16 balance, weight, distance, and conflict-weight problems and less than 3 correct on the 8 conflict-distance and conflict-balance problems; stage 3 requires at least 10 correct on the 12 balance, weight, and distance problems and fewer than 10 correct on the 12 conflict problems; stage 1 requires at least 10 correct on the 12 balance, weight, and conflict-weight problems and fewer than 3 correct on the 12 distance, conflict-distance, and conflict-balance problems. Stage 2 is given scoring priority over Stage 3 because the criteria for Stage 2 are more specific, particularly on how to score conflict-weight problems.

Addition The Addition test set helps to distinguish a genuine torque rule from a mere addition rule. It contains all 148 addition problems (among all conflict problems), a few of which may be included in the training set when expanding by one pattern per epoch. Typically, no more than one or two such patterns get included in the train set.

Torque The torque test set contains the 27 torque problems not randomly selected to expand the training set in torque-rule training. Among all conflict problems, there are 52 torque problems, 25 of which are used in training. There is a small probability that these problems are selected when expanding the train set by one pattern per epoch, but in practice no more than a single pattern is included in this way. If a network gets all (or nearly all) torque problems correct, then it is diagnosed as following a genuine torque

rule as opposed to solving balance-scale problems with the often successful addition rule.

Procedure

We ran 20 network systems, with each system containing each of the three described networks, which were tested on each of the three test sets every 25 epochs. An epoch is a pass through the entire training set.

Results

Intuitive networks trained for a mean of 399 epochs (SD = 44), torque-rule networks for a mean of 291 epochs (SD = 86), and confidence networks for a mean of 1103 epochs (SD = 161). Torque-rule networks and confidence networks train until all outputs are within the score-threshold of targets, whereas training of intuitive networks is stopped early (as noted) in order to mimic stage 3 performance. Confidence networks take longer because they are learning to approximate a continuous function (with a smaller score threshold) as opposed to the classification task (with two binary outputs) for the torque and intuitive networks.

Figure 1 presents mean stage classification on the Siegler-TD test set for 20 intuitive networks over epochs. Performance at stage 1 is evident at epoch 25, stage 2 at epochs 75-150, and stage 3 at epochs 200-350. Epoch 50 marks the transition between stages 1 and 2, and epoch 175 marks the transition from stage 2 to 3. Thus, these networks capture the first three balance-scale stages seen in children from about five years of age up through early adolescence.

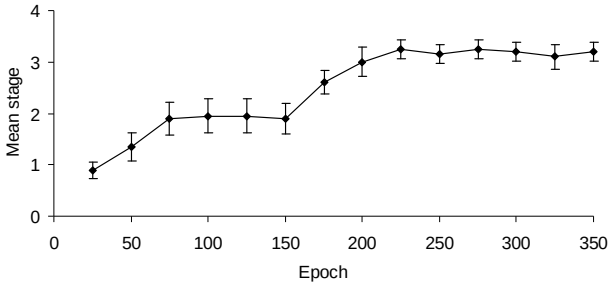

Figure 1: Mean stages on Siegler-TD test set of 20 intuitive networks over epochs ±SE.

Figure 2 shows accuracy, in terms of mean proportion correct, in 20 intuitive networks on each of the three test sets over epochs. This confirms that intuitive networks learn to perform well on the Siegler-TD and Addition test sets, but not on the Torque test set. With this kind of training regimen, lacking sufficient experience with torque problems, something else is clearly required to achieve successful performance on problems that can be solved only by comparing torques.

The extra factor in this simulation that allows networks to succeed on torque problems is the neurally-coded torque-rule network, mimicking the instruction that many children receive in secondary-school science classes, along with enough examples of torque problems to practice on, and a confidence network that allows selection of the torque-rule network when confidence in the intuitive solution is low.

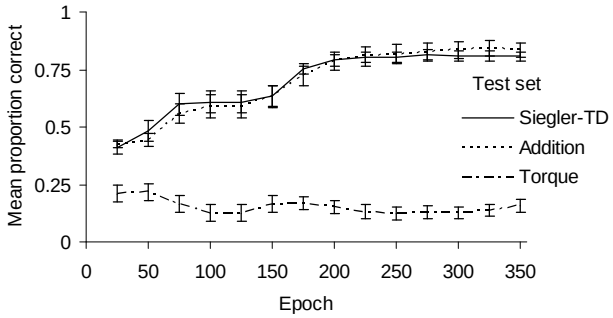

Figure 2: Accuracy of 20 intuitive networks on three test sets over epochs ±SE.

Figure 3 presents mean accuracies of the two network modules on the three test sets after training is complete, along with 95% confidence intervals. Although intuitive networks perform well on the Siegler-TD and Addition test sets, they do badly on Torque problems. In contrast, combining the two network modules, by using the intuitive network when it provides an answer in which the confidence network is sufficiently confident and resorting to the torque network when confidence is low, allows a high level of performance on all three test sets.

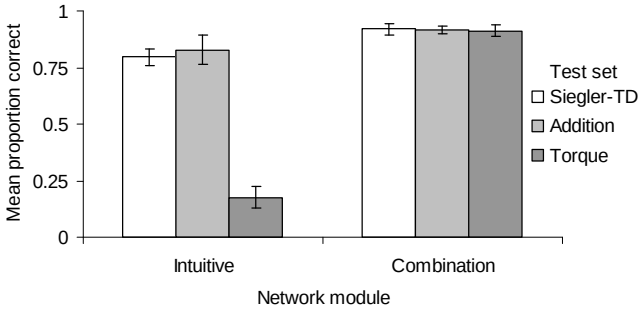

Figure 3: Accuracy of network modules on three test sets after training ±95% confidence interval.

The mean proportions of problems solved by the intuitive network under combination conditions were .60, .47, and .09 on the Siegler-TD, Addition, and Torque test sets, respectively. That is, torque problems are almost never solved intuitively, whereas about half of non-torque problems are solved intuitively.

Discussion

Our results show that, contrary to previous claims (Quinlan et al., 2007), it is indeed possible for neural networks to learn to follow a genuine torque rule in balance-scale development. In this context, a torque rule can be considered genuine if it generalizes very well to conflict problems that cannot be solved a simpler rule that merely adds, rather than multiplies, weight and distance

information. Our networks generalize correctly to such untrained torque problems with over 90% success.

Previous work showed that it was possible for neural networks to acquire a genuine torque rule from examples alone or by recruiting an injected torque rule if sufficient torque problems were provided in training (Shultz et al., 2007). But these models did not progress through the first three stages of balance-scale performance seen in children. Progression through these earlier stages requires a training set with random problem selection subject to a strong bias in favor of equal-distance problems in which there are the same numbers of weights placed equally distant from the fulcrum, thus preventing sufficient experience with the very rare torque problems.

The present model is the only neural-network system to so far demonstrate progression through all four balance-scale stages finishing with a genuine torque rule. Our multi-network system captures the first three balance-scale stages with an intuitive network that learns only from examples. Then a knowledge-based network with an injected torque rule in the source-knowledge pool and additional torque training examples builds on this early intuitive training by recruiting this taught torque rule and learning how to use it.

Just as with secondary-school science students, a lesson on torque does not guarantee a torque solution. The taught torque rule must be stored, recruited, and practiced; and even then it may not be used on simple balance-scale problems that can be solved intuitively. A confidence network learns to predict whether the intuitive network is able to solve a given balance-scale problem based in part on how close the torques are on each side. If confidence in the intuitive network is too low, the torque-rule network provides a more accurate answer.

Our model is consistent with psychological evidence that response time is slower with increasing age and increasing rule complexity (van der Maas & Jansen, 2003). This is because using the torque-rule network, the time for which adds to that of using the intuitive network, is more likely with increased training and more difficult problems. It is also consistent with the intuition that ordinary people do not invent a torque rule on their own, but rather are sometimes taught about torque in science classes (Siegler, personal communication). Our model predicts, perhaps uniquely, that response times would increase on problems with small absolute torque differences between the sides of the scale.

The ability of KBCC to incorporate differentiable functions into its source knowledge pool is a promising and novel way to integrate neural-network and symbolic approaches to cognitive modeling. Our use of a confidence network indicates that neural networks may be able to simulate aspects of meta-cognition.

Acknowledgments

This research is supported by a grant to TRS from the Natural Sciences and Engineering Research Council of Canada.

References

Baluja, S., & Fahlman, S. E. (1994). *Reducing network depth in the cascade-correlation learning architecture.* (No. Technical Report CMU-CS-94-209). Pittsburgh, PA: School of Computer Science, Carnegie Mellon University.

Boom, J., Hoijtink, H., & Kunnen, S. (2001). Rules in the balance: Classes, strategies, or rules for the balance scale task. *Cognitive Development, 16*, 717-735.

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.

McClelland, J. L. (1989). Parallel distributed processing: Implications for cognition and development. In R. G. M. Morris (Ed.), *Parallel distributed processing: Implications for psychology and neurobiology* (pp. 8-45). Oxford, UK: Oxford University Press.

Quinlan, P. T., van der Maas, H. L. J., Jansen, B. R. J., Booij, O., & Rendell, M. (2007). Re-thinking stages of cognitive development: An appraisal of connectionist models of the balance scale task. *Cognition, 103*, 413-459.

Schmidt, W. C., & Ling, C. X. (1996). A decision-tree model of balance scale development. *Machine Learning, 24*, 203-229.

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., Mareschal, D., & Schmidt, W. C. (1994). Modeling cognitive development on balance scale phenomena. *Machine Learning, 16*, 57-86.

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.

Shultz, T. R., & Takane, Y. (2007). Rule following and rule use in simulations of the balance-scale task. *Cognition, 103*, 460-472.

Siegler, R. S. (1976). Three aspects of cognitive development. *Cognitive Psychology, 8*, 481-520.

Thivierge, J. P., Dandurand, F., & Shultz, T. R. (2004). Transferring domain rules in a constructive network: Introducing RBCC. In *Proceedings of the IEEE International Joint Conference on Neural Networks* (pp. 1403-1409).

van der Maas, H. L. J., & Jansen, B. R. J. (2003). What response times tell of children's behavior on the balance scale task. *Journal of Experimental Child Psychology, 85* 141-177.