Connectionist Models of Reinforcement, Imitation, and Instruction in Learning to Solve Complex Problems

Frédéric Dandurand, *Member, IEEE*, and Thomas R. Shultz

***Abstract*—We compared computational models and human performance on learning to solve a high-level, planning-intensive problem. Humans and models were subjected to three learning regimes: reinforcement, imitation, and instruction. We modeled learning by reinforcement (rewards) using SARSA, a softmax selection criterion and a neural network function approximator; learning by imitation using supervised learning in a neural network; and learning by instructions using a knowledge-based neural network. We had previously found that human participants who were told if their answers were correct or not (a reinforcement group) were less accurate than participants who watched demonstrations of successful solutions of the task (an imitation group) and participants who read instructions explaining how to solve the task. Furthermore, we had found that humans who learn by imitation and instructions performed more complex solution steps than those trained by reinforcement. Our models reproduced this pattern of results.**

***Index Terms*—Cognitive science, learning systems, neural networks, problem-solving.**

I. Introduction

SOLVING problems is important for biological and artificial agents alike to survive and adapt in ever changing environments. Agents can learn to solve problems by trial-and-error, but this process is often inefficient and may expose agents to fatal errors. An important evolution in biological agents has been the ability to learn from observation and instructions of other agents. Agents who can learn from peers have an adaptive edge because they do not need to "reinvent the wheel." Imitation is perhaps the most universal mechanism for social learning because mentors do not need to be aware, or even willing in competitive settings, to play such a role. Most other social learning mechanisms such as teaching, giving hints, etc., require mentors who have the time, motivation, and ability (e.g., pedagogy) for transmitting knowledge, which may be rare in many environments.

Manuscript received December 13, 2008; revised August 24, 2009. First published August 28, 2009; current version published October 21, 2009. The work of F. Dandurand was supported by a McGill Major scholarship. The work of T. R. Shultz was supported by the Natural Sciences and Engineering Research Council of Canada.

F. Dandurand was with the Department of Psychology, McGill University, Montreal, QC H3A 1B1, Canada. He is currently with the Laboratoire de Psychologie Cognitive, CNRS & Université de Provence, UMR 6146 Pôle 3 C, 3, place Victor Hugo, 13331 Marseille Cedex 3, France.

T. R. Shultz is with the Department of Psychology and School of Computer Science, McGill University, Montreal, QC H3A 1B1, Canada.

Digital Object Identifier 10.1109/TAMD.2009.2031234

Traditionally search, heuristics, hints, induction, and reasoning by analogy were among the main research interests in problem solving [1]–[4]. Most computational models of problem solving (e.g., SOAR: [3], ACT-R: [5], [6]) reflect these emphases and typically involve symbolic processing, although some recent work has introduced more biologically plausible reinforcement-learning models [7], [8]. Most work on problem solving is grounded in information processing theory [9] and, despite the adaptive value of imitation, has shown comparatively little interest in studying and modeling how people learn to solve problems by watching demonstrations. In fact, imitation learning of problem solving was often dismissed as rote memorizing [10], which contrasts with the current views of imitation as being cognitively complex and as possibly involving understanding of intentions and goals, e.g., [11].

Imitation learning is currently an active research topic in cognitive and social psychology. Most current work on imitation learning focuses on motor tasks and there is comparatively little work on high-level cognitive tasks like problem solving; one notable exception is a Bayesian model of imitation [12].

In the current paper, we present connectionist simulations of psychological data presented in Dandurand *et al.* [13] on learning in a planning-intensive problem we call the Gizmo Problem Solving Task. We model three kinds of learning regimes: imitation, instruction and reinforcement. We ask three questions as follows. First, can neural-network-based computational models be trained to solve gizmo problems under these learning regimes? Second, do these models reproduce the pattern of human performance in the gizmo task? Third, how do models compare in learning efficiency? We compare humans and models on their solution accuracy, and on the simplicity and symmetry of the actions selected.

Computational models of problem solving are largely dominated by symbolic approaches. Whereas associative learning is fundamental to neural network processing, symbolic processing is primarily based on search [3]. Nevertheless, symbolic models can account for some learning in problem solving. For instance, the ACT-R production system was used to model how experience and practice affected planning in the Towers of Hanoi problem [6]. With experience, human participants gradually increased their tendency to plan longer sequences of moves, with the optimal strategy gradually replacing a less effective though easier one, and an ACT-R model reproduced these data. Another rule-learning system, SOAR, like humans, improves in speed and processing efficiency with practice [3]. SOAR production rules are learned by summarizing the results

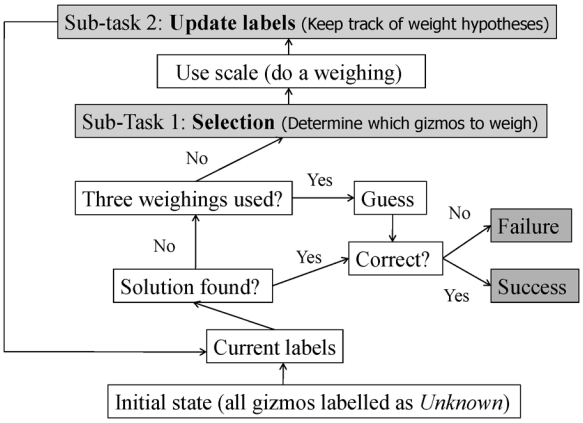


Fig. 1. Task analysis of the gizmo problem solving task.

of look-ahead search. Subsequent encounters with similar conditions can activate those productions, thus replacing search with memory access.

Similarly, most of the work in modeling instruction learning has been done in symbolic systems, except for a few connectionist models [14]. Recent work has investigated how advice giving can be modeled within an extended reinforcement learning framework [15].

Modeling instruction learning in neural networks faces two important challenges. First, how can instructions be represented in neural terms? Second, how does the system use those instructions to solve the task at hand? Our knowledge-based neural network model of instruction learning offers a possible answer to the first question: instructions are encapsulated into pretrained neural networks capable of being recruited. Our model also partially addresses the second question. If a rule correlates well with residual network error, it may be recruited.

Research in human problem solving can greatly benefit from modern computational modeling techniques in which learning is central. There have already been some successful neural-network models of problem solving, e.g., [16], [17]; see [18] for a review. Here, we show how neural networks can be usefully combined with the classical search-space approach to problem solving, taking full advantage of the superior learning and generalization ability of these networks.

A. The Gizmo Problem Solving Task

We compared performance of human and artificial (computational model) problem solvers on a complex, well-defined problem called the gizmo problem solving task. The task consisted in finding, with three uses of a balance scale, the odd gizmo in a set of twelve gizmos. Gizmos had uniform weights except for the target gizmo, which could be either heavier or lighter than the other gizmos. The goal was to reliably detect the target gizmo and its relative weight in any of the 24 possible cases (12 gizmos $\times$ 2 weights). The target could only be identified based on its weight, as all gizmos looked exactly alike, and thus could not be visually distinguished.

As shown in Fig. 1, solving this problem involved an alternation between two subtasks. First, problem solvers needed to select gizmos, that is, to decide which gizmos to weigh and where to install them on the balance scale. Second, problem solvers updated hypotheses they made about possible gizmo weights, after seeing the scale result, by labelling gizmos. Problem solvers alternated between selecting gizmos and updating labels until they found a solution or until they used the scale three times. An optimal solution is presented in tree form in Fig. 2.

This task belongs to a class of so-called *well-structured* problems characterized by their clear initial and goal states, and by their precisely defined operators and constraints [19]. A simpler variant of this task, called the Coin problem, has been previously used in a psychological experiment. This variant involved finding a lighter target object (a coin) among a set of eight coins using at most two weighings on a balance scale [20]; see [21] for a mathematical discussion of this class of problems.

B. Comparing Performance

To compare performance of humans and models on this task, we first measured solution accuracy, defined as the proportion of correct answers on trials. Second, we measured complexity and asymmetry of the actions selected [22]. Optimal solutions required complex and asymmetrical arrangements of gizmos (see Fig. 2 and Fig. 4). Human problem solvers seldom spontaneously found an optimal solution, possibly because striving for simplicity is a universal and pervasive principle across the cognitive spectrum from low-level perception to high-level cognition [23]. A related principle is the symmetry bias found, for instance, in perception [24].

To measure complexity (the inverse of simplicity), we summed the total number of labels present on each side of the scale. Seven labels are available: unknown (U: heavy, light, or normal), heavy or light (HL), heavy or normal (HN), light or normal (LN), heavy (H), light (L), and normal (N). In simulations, we drop the heavy or light (HL) label because humans almost never used it. To measure asymmetry, we counted the total number of differences in labels between left and right sides of the scale, i.e., whenever a label was present on one side of the scale but not on the other, one unit of asymmetry was added. Table I shows examples of complexity and of asymmetry measures.

The upper bound of complexity was 12, when items of each of the six label categories (U, LN, HN, L, H, and N) were installed on both sides of the scale. The upper bound of asymmetry was six when items of each category were installed on the scale without a matching element on the other side.

We expected to measure relatively low complexity and asymmetry scores in human solutions. Comparing detailed solution steps was difficult because of the cascading effect of weighings—a small difference in the selection of gizmos at the first weighing may have yielded a large difference in problem configuration at the third weighing. Likewise, we did not model errors because human errors were rare and concerned misuse of the GU interface (GUI) or distractions, having no apparent connection to problem solving strategies.

C. Human Performance on the Gizmo Task

In a previous laboratory experiment [13], we measured human performance on the gizmo task. Participants were

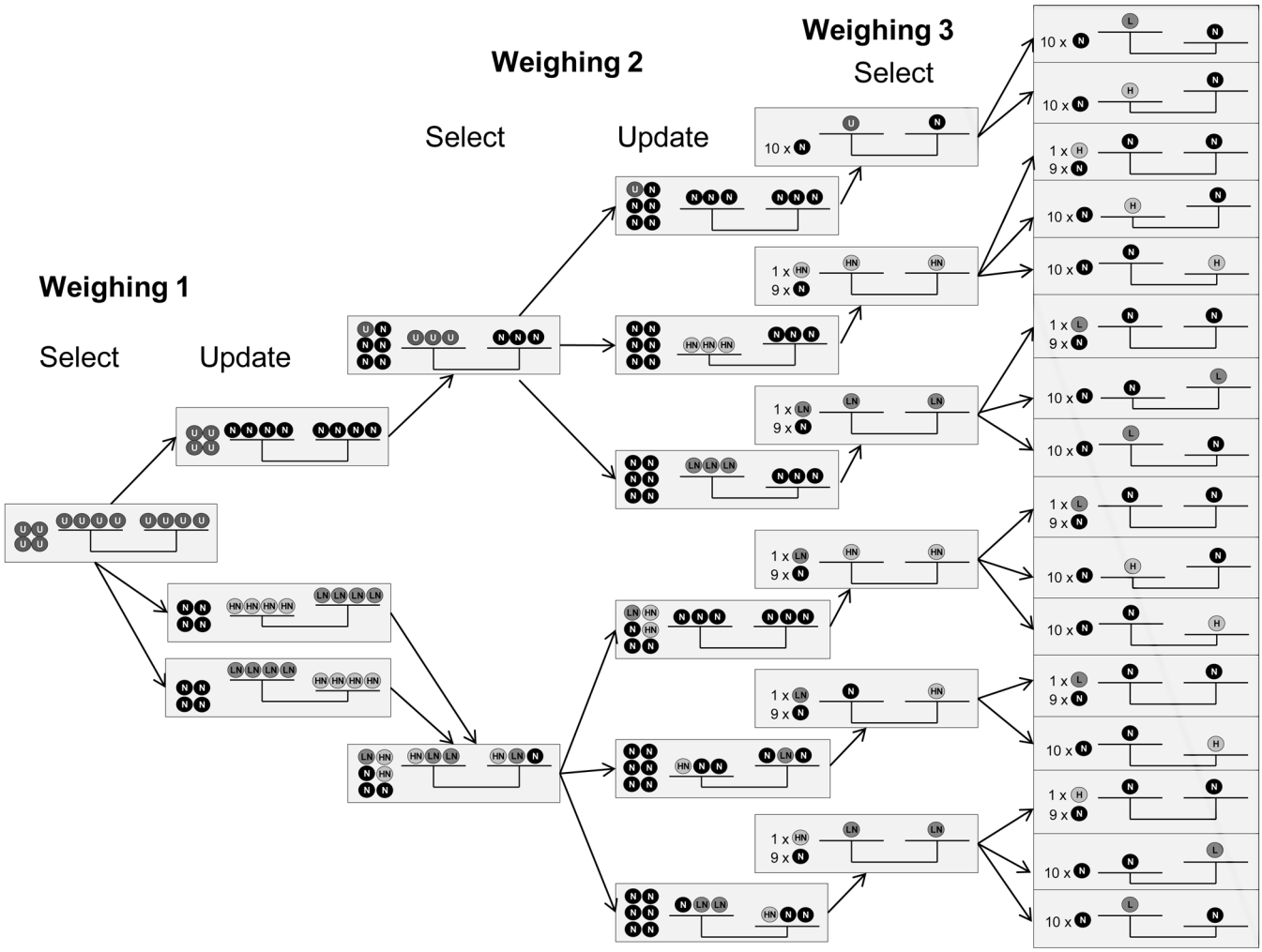


Fig. 2. An optimal solution to the gizmo problem solving task. Five demonstrations were shown to the imitation group participants. A demonstration consisted of a randomly selected, complete branch (three weighings) leading to one of the leaves. Only 16 of the 24 leaves are shown because the branches can be grouped together at Weighing 2. The nine gizmo selection rules can be found in the columns labeled as "Select"; one rule for Weighing 1, two for Weighing 2, and six for Weighing 3.

TABLE I
EXAMPLES OF MEASURES OF COMPLEXITY AND ASYMMETRY

	Example	Index
Complexity	HN HN (1) vs. N N (1)	2
	HN LN LN (2) vs. HN LN N (3)	5
Asymmetry	HN HN vs. N N	2
	LN LN LN vs. LN LN LN	0
	HN LN LN vs. HN LN N	1

McGill undergraduate and graduate students. The 68 participants yielded 63 (17 males and 46 females) usable data samples (21 per experimental group). Participants were excluded if they could not finish the warm-up task within 30 min $(\mathrm{n} = 3)$, or if they were identified as statistical outliers on a q-q plot $(\mathrm{n} = 2)$. Participants were randomly assigned to experimental groups. The chance to win a $50 prize encouraged maximal performance by keeping participants motivated.

Participants first performed a warm-up task to familiarize with the use of the program. Second, some participants had a learning session (details below). Finally, participants solved gizmo problems for 30 minutes, or until they successfully solved a complete block of 24 different trials consecutively. They completed an average of 18.6 trials (min: 6, max: 37, $\mathrm{std\ dev} = 7.9$). A different combination of gizmo and weight (light or heavy) was randomly selected for each trial from a list of unsolved trials. When an error was made, the list was reset back to the complete block of 24 trials. Participants got a different trial each time, regardless of whether they successfully resolved the previous one.

Participants could install any combination of gizmos (up to 12 on each side of the scale) on any given weighing. We explicitly instructed participants to keep track of hypotheses they made about gizmo weights based on information obtained using the scale result (balanced, left heavier, or right heavier). To enforce this labeling operation, gizmos were shuffled when the scale was activated, so they could not be identified by their location, only by their label. A screen shot from the program used for the experiment[1] is presented in Fig. 3.

[1]Available online at: http://lnsclab.org/html/BallsWeightExperiment/PlayVersion/play.html.

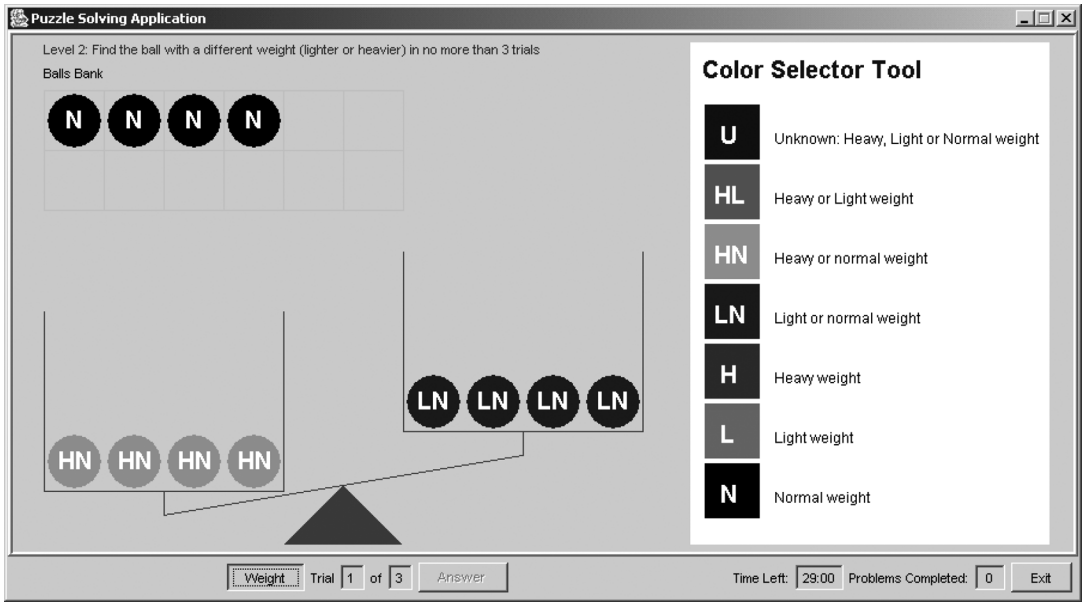


Fig. 3. Screen shot of the gizmo problem solving task used in the laboratory experiment. Participants were asked to find, with three uses of a scale, the one gizmo that was either heavier or lighter than the rest of a set of twelve gizmos. The screen was divided in three sections: the gizmo bank (on the top left), the balance scale (at the bottom, to the left), and the "Color Selector Tool" (on the right). Participants were instructed to use labels available in the color selector tool to keep track of their hypotheses about possible weights of gizmos. In this example, we see the first weighing in which four gizmos have been determined to be of Normal weight, four of Heavy or Normal weight, and four of Light or Normal weight. Any number of gizmos (up to 12) could be installed on either side of the scale, as gizmos can be piled or stacked on platters.

We manipulated the conditions under which participants learned to solve gizmo problems. Participants were randomly assigned to one of three learning groups: 1) in a reinforcement learning group, they were simply told if their answers were correct or not at the end of each trial, 2) in an imitation learning group, they watched five problem-solving demonstrations (different gizmo/weight combinations), and 3) in an instruction learning group, they studied for 10 min verbal instructions presented as if-then rules (see Fig. 4).

Demonstrations presented to participants in the imitation learning group were five different, randomly selected branches of the solution tree (see Fig. 2). The complete set of instructions is presented in Fig. 4. Participants in the verbal instruction learning group were presented with subsets of the complete instructions in which information content was matched with five random demonstrations. In other words, the instruction group was a yoked control for imitation, where each subset of instructions matched a randomly selected set of demonstrations.

We found that participants who watched demonstrations or read instructions were more accurate than participants who were simply given feedback about their answers [13]. Accuracy results are presented in Table II.[2]

II. Methods

We present computational models of the first subtask, gizmo selection, assuming an optimal agent to provide label updates. First, we justify the use of such an optimal agent for label updates. Second, we describe the common element of our models: cascade-correlation neural networks. Finally, we introduce the details of our computational models.

A. Optimal Agent for Label Updates

A detailed analysis of all labeling operations ($\mathrm{N} = 3444$) provided support for using such an optimal, or ideal, agent. Humans provided correct labels 95.15% of the time. Incorrect or inconsistent uses of labels amount to less than 3.0%, and forgetting or neglecting to update labels for which additional information is available, less than 1.9%. Incorrect labeling did not necessarily reflect a failure of the labeling strategy. Many labeling errors could be explained by errors in GUI-interface manipulation and inattention. Furthermore, participants did not always overtly label gizmos despite being instructed to. Yet, in many cases the logic of their solutions strongly suggests they mentally kept track of the gizmo weights.

B. Cascade-Correlation Neural Networks

Cascade-correlation [25] is a constructive neural network algorithm in which computational units, generally sigmoid units, are recruited as necessary to solve a task. We chose cascade-correlation because this family of algorithms has successfully been used to model a number of learning and cognitive-developmental tasks [26]. For this research, we used a variant called sibling-descendant cascade-correlation (SDCC) [27] which has the interesting property of limiting network depth by eliminating some of the cascading weights of classical cascade-correlation. Cascading weights may have little impact on performance, including generalization [28].

Cascade-correlation learns by alternating between input and output phases. In input phases, computational units (e.g., with sigmoid activation functions) in a recruitment pool are trained

[2]In these group accuracies, participants are equally weighted regardless of their speed or the number of trials they completed. In contrast, previously reported accuracies were averaged across all trials in each group, giving more weight to faster participants [13]. Regardless of weighting method, the pattern of results was the same.

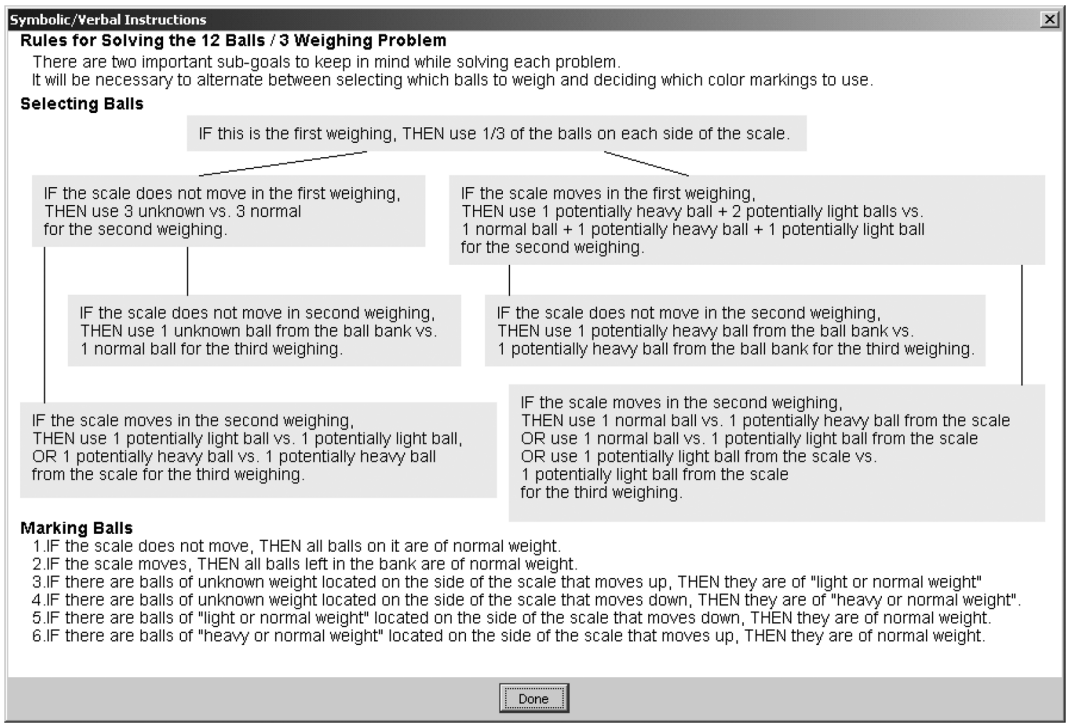


Fig. 4. A complete set of instructions to solve the gizmo problem solving task with the same solution as shown in Fig. 3. A subset of those instructions was selected to match the information content of each set of demonstrations.

TABLE II
ACCURACIES OF HUMAN PARTICIPANTS IN A GIZMO PROBLEM SOLVING EXPERIMENT, EXPRESSED AS MEAN PROPORTION CORRECT

Experimental group	Accuracy	
	Mean	Standard deviation
Reinforcement learning	0.58	0.18
Imitation learning	0.72	0.27
Instruction learning	0.69	0.22

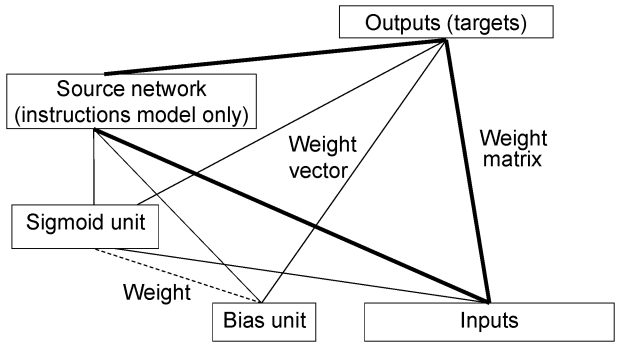


Fig. 5. Sample KBCC network. It has recruited a single sigmoid hidden unit followed by a source function. Thick solid lines represent connection-weight matrices, thin solid lines represent connection-weight vectors, and the dashed line represents a single connection weight. All our models could recruit sigmoid units, but source networks were only available to the instruction learning models.

to maximize covariance with residual network error. At the end of an input phase, when covariance stops increasing, the unit with the highest covariance is inserted and connected to the current network structure. In standard cascade-correlation [25], units are always installed on new hidden layers, creating deeply cascaded networks. In contrast, the sibling-descendant cascade-correlation (SDCC) variant chooses to install units on the current deepest hidden layer (sibling units) or on a new layer (descendant units), again by selecting the candidate whose activations track network error most accurately. To limit network depth, SDCC typically slightly penalizes descendant units. In output phases, connection weights feeding the output units are trained to minimize network error using an algorithm for training feed-forward networks such as QuickProp [29].

Knowledge-based cascade-correlation (KBCC) [30] is another extension of cascade-correlation that allows for prior knowledge to be included in the recruitment pool and to compete with simple units. Prior knowledge can be previously trained networks or neurally-implemented rules [31]. Recruitment criteria are the same for complex prior knowledge modules as for simple logistic units: CC selects the candidate whose output values best co-vary with residual network error. KBCC allows networks to recruit existing knowledge in the service of new learning. It is used here to model the verbal instruction group by including in the knowledge pool SDCC networks pre-trained on each of a set of rules (instructions) for selecting gizmos. Fig. 5 presents a sample KBCC network. SDCC networks are built in a similar fashion, except they recruit sigmoid units only.

C. Input and Output Encoding

Because gizmos were shuffled when the scale was activated, participants could not identify gizmos based on their location. They could only distinguish gizmos by their labels. Similarly,

in our simulations, gizmos were distinguished based on their labels, not individually.

States were encoded as the proportion of gizmos with each label. For example, six gizmos were coded as $6/12 = 0.5$. Six units coded the proportion of gizmos of each label type in order: U, HN, LN, H, L, N.

Actions were encoded as the proportion of gizmos of each label type to install in each container (bank, left side of scale, and right side of scale) using 18 units (6 labels $\times$ 3 containers). For example, if the twelve gizmos were labeled as Unknown, and the selection action consisted in weighing six gizmos on the left side of the scale $(6/12 = 0.5)$ against six gizmos (0.5) on the right side, leaving no gizmo in the bank, the action was encoded as: 0 0 0 0 0 0; 0.5 0 0 0 0 0; 0.5 0 0 0 0 0. When computing proportions, the denominator was the total number of gizmos (that is, always twelve).

D. Model Details

In this section, we present details of the three models used in our simulations.

1) Reinforcement Learning: First, we modeled the reinforcement learning group using an on-policy temporal-difference (TD) learning technique called SARSA [32], previously used in [33]. SARSA was named after the quintuple that the algorithm uses $(\mathrm{s_t}, \mathrm{a_t}, \mathrm{r_{t+1}}, \mathrm{s_{t+1}}, \mathrm{a_{t+1}})$:

$$\mathrm{Q}(\mathrm{s_t}, \mathrm{a_t}) \leftarrow \mathrm{Q}(\mathrm{s_t}, \mathrm{a_t}) + \alpha\left[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - \mathrm{Q}(\mathrm{s_t}, \mathrm{a_t})\right] \tag{1}$$

where Q is the expected or predicted value (quality) of reward, s is a state, a is an action, r is a reward, and indices t, and $\mathrm{t}+1$ are used for current and next states and actions, respectively; a is a learning rate (here, we use 0.1), and γ is a discount factor (here, 1.0). This last value effectively means no discounting. In other words, solutions were equally rewarded for their accuracy regardless of whether they used the maximum number of weighings (here 3) allowed, or they used fewer weighings. This choice was motivated by the fact that participants did not appear to actively attempt to achieve solutions involving fewer than three weighings. In fact, humans produced short solutions in only about 10%[3] of trials. Second, the problem description made no mention that shorter solutions should be preferred. In fact, correct solutions to this problem require the maximal number of weighings allowed (see Fig. 2). Shorter solutions occur in cases of fortunate guesses, and are thus unreliable.

SARSA was chosen for its neurological plausibility. Some brain structures (e.g., basal ganglia and the striatum) may well learn using a SARSA-like mechanism [34]–[36]. Our SARSA-based model learned to accurately predict the reward resulting from taking an action (that is, a selection of gizmos to install on the scale) in a given state (that is, the current labeling of gizmos). Lookup tables are typically used to exhaustively store expected reward values for encountered states and actions. In contrast, we used SDCC neural networks as function approximators mapping concatenations of states and actions as inputs

[3]This value includes GUI manipulation errors.

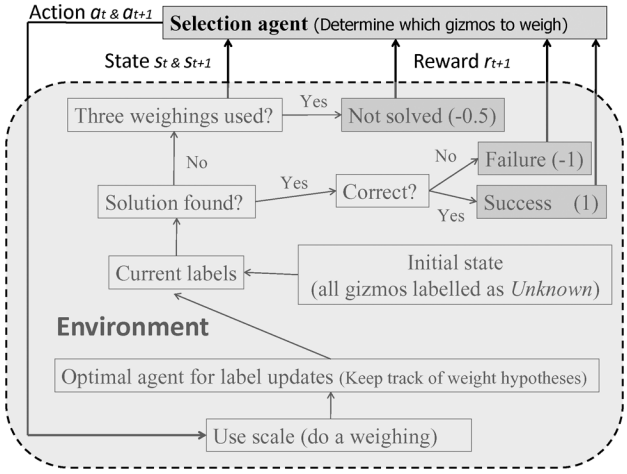


Fig. 6. Reinforcement model showing agent-environment interactions. From the perspective of the agent making selection decisions, the optimal agent performing label updates is part of the environment.

onto the expected reward for taking particular actions from particular states. These neural-based function approximators generalize better to unvisited states and actions, and are more neurologically plausible than lookup tables.

The model improved initial inaccurate reward estimates by trial-and-error exploration of the problem search space. Our experimental setup represented a typical scenario in which reinforcement learning occurred based on impoverished and infrequent information. Learners were only told if their answers were correct or not (i.e., given a binary reward) in terminal states. Learners did not get any rewards for gizmo selection actions performed in non-terminal states, that is, for the first and second weighings. More specifically, rewards were as follows: $+1$ for a correct answer, -1 for an incorrect answer, -0.5 when the system did not give any answer after three weighings, and 0 for nonterminal states (i.e., no reward). Because TD-learning implements a form of temporal back-propagation of rewards to states and actions that lead to the reward, agents could eventually learn accurate estimates of states and actions associated with the first and second weighings despite receiving no rewards for those.

Similar to previous models of the reinforcement learning condition [33], computational models of the selection subtask selected only actions that had the same number of gizmos on both sides of the scale because humans installed an equal number of gizmos on the two platters of the scale in 98.6% of their actions. Fig. 6 shows the reinforcement learning framework for this problem. The elements are essentially the same as those of the task analysis for humans presented in Fig. 1. The only minor difference between humans and models resides in the "Not solved" terminal state for which a reward was directly generated for models without the need to guess. Also note that because the agent for providing label updates was performing independently and outside the control of the selection agent, it could be considered as part of the environment as far as the agent learning selections was concerned.

Our SDCC networks had 24 inputs: six to code the state concatenated with 18 to code the action. They had a single output

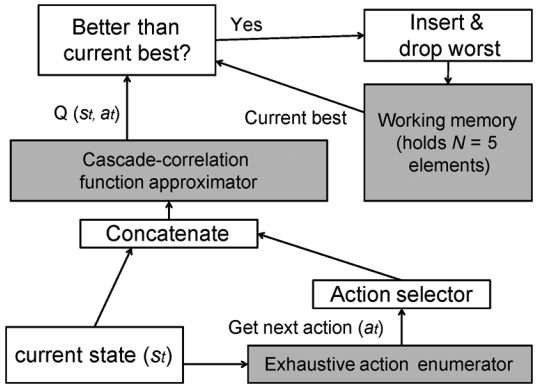


Fig. 7. Architecture of the selection agent. See Fig. 5 for an illustration of the cascade-correlation function approximator.

coding the expected reward value from -1 to $+1$, and the output sigmoid range was adjusted accordingly. To select actions to take, we used a modified Softmax technique. We truncated the number of alternatives considered to the five actions that have the highest expected reward. We had previously found five to yield good performance [33], and it is compatible with estimates of human working memory size [37], [38]. An illustration of the computational model architecture is given in Fig. 7 which essentially expands the internal functioning of the selection agent in Fig. 6.

2) Imitation Learning: We modeled the imitation learning condition with SDCC neural networks using demonstrated states as inputs and demonstrated actions as targets or outputs. Models were trained with five demonstrations of three weighings, matching what participants watched in the imitation learning condition. Because each weighing corresponded to a different pair of state and action coded as one training pattern, the training set contained 15 patterns (5 demonstrations $\times$ 3 weighings). SDCC networks had 6 inputs coding the demonstrated state and 18 outputs coding demonstrated actions. We used the same coding for states and actions as for the reinforcement model just described. The recruitment pool included only sigmoid units: four on the current deepest layer (sibling) and four on a new cascaded layer (descendant), with each unit having different random initial conditions. An illustration of the computational model architecture is given in Fig. 5—note that only sigmoid unit could be recruited, there were no prior knowledge source networks in the models of imitation learning.

3) Instruction Learning: We modeled the instruction learning condition using KBCC networks. While this model resembled the imitation learning model (same task, inputs, outputs and training data), the key difference resided in the recruitment pool. The recruitment pool of the instruction learning model included, in addition to sigmoid units, networks pre-trained on all nine selection rules (see Fig. 2 under "Select" columns). There was one selection rule for the first weighing, two rules for the second one, and six rules for the last weighing. Each rule was pre-trained in a SDCC network using a training set containing a single pattern corresponding to the rule. Note that networks saw the full set of rules, whereas humans only saw a subset. We were interested in whether networks could spontaneously focus on the subset of rules relevant for the problems they saw. As is customarily done with KBCC, we also included a control for complexity. Each candidate rule had a matching random network with the same topology but with connection weights reset to random values to make sure rules were recruited for knowledge, not mere complexity. As a result, the recruitment pool for models of instruction learning contained a total of 148 candidates: two sigmoid candidates as siblings, two sigmoid candidates as descendants, four candidates of each of the nine rules (two as siblings, two as descendants; $\mathrm{total} = 4 \times 9 \times 2 = 72$), and for each rule network, a matching random network. An illustration of the computational model architecture is given in Fig. 5 where source networks consisted of selection rules and complexity-matched controls.

As previously mentioned, demonstrations consisted in randomly selected branches of the tree presented in Fig. 2. For each demonstration, the three rules traversed by the demonstrated branch, one for each weighing, were necessary for solving the problem. Thus which of the nine rules were necessary (and therefore may be expected to be recruited) depended on the random selection of demonstrations. As more demonstrations were presented, additional demonstrations tended to become more redundant and less informative. For instance, all demonstrations show the same first weighing rule consisting of a four versus four gizmo selection. Consequently, we expected diminishing returns as more demonstrations were presented.

E. Simulation Parameters

In CC networks (SDCC and KBCC alike), score threshold controls how close network output values must be to target values for learning to be successful. In output phases, training finishes when, for all outputs $\mathrm{o_{i,j}}$ and targets $\mathrm{t_{i,j}}$, $|\mathrm{o_{i,j}} - \mathrm{t_{i,j}}| < \mathrm{score\ threshold}$. In reinforcement models, we set the score threshold to 2.5% of the reward range, that is, 0.05. In imitation and instructions models, the outputs encoded proportions of gizmos, and thus range between 0 and 1, by increments of 1/12 (because there were twelve gizmos). We empirically found that tolerating a distance between the outputs and the targets of 0.4 times those increments gave satisfying results, and thus score threshold was set to $0.4 * 1/12 = 0.033$; see Appendix for details.

Other cascade-correlation parameters were set as follows. No weight change was allowed to be greater in magnitude than maximum growth factor times the previous step for that weight [29]. Here, maximum growth factor was set to 2.0. CC changes phase if error reduction (or covariance increase) has been lower than change threshold across patience epochs. Here we used a patience of eight epochs and change thresholds of 0.01 in output phases and 0.03 in input phases. CC also switches phase after having reached maximum epochs in the current phase, set here to 100. The decay parameter, set to 0.0002 in output phases and to 0 in input phases, was used to keep weights from growing too big. Finally, the learning rate, which controls the amount of gradient descent used in updating weights, was set to 0.175 in output phases and to 1.0 in input phases.

F. Testing Technique

To test and evaluate models, we used our computational models to select gizmos (first subtask) and used an optimal

agent to update gizmo labels (second subtask). Our test software coordinated the two subtasks (see Fig. 1), provided appropriate inputs to models and agents, read their outputs to determine actions and label updates to perform, and kept track of gizmo locations and the current state. For imitation and instruction models, the current state was presented as the input and the selected action was read at the output. For reinforcement models, all valid actions from the current state were scored, and the selection action with the highest expected reward was always chosen in test mode (Hardmax[4]). The optimal agent updated gizmo labels based on the following: 1) the current state, 2) the gizmo selection that the model made, and 3) the scale state result (balanced, right heavier, or left heavier). Because CC function approximators output continuous values, we rounded these values to the closest valid proportion. For example, if a node representing a proportion of gizmos out of twelve outputs a value of 0.238, the testing system would decide this represented an action involving three gizmos because $3/12 = 0.25$ is the closest valid proportion value.

The test system verified answers given for each of the twenty-four different problems (12 gizmos $\times$ 2 weights) and computed accuracy as the number of correct answers out of twenty-four. An answer was correct if, after the third weighing, all gizmos were labeled as Normal weight, except the target gizmo correctly labeled as Heavy or as Light. Note that although imitation and instruction models were trained respectively using five problem demonstrations or a corresponding set of instructions, they were tested on all twenty-four possible problems. This allowed us to evaluate how models generalize what they have learned to problems unseen in training.

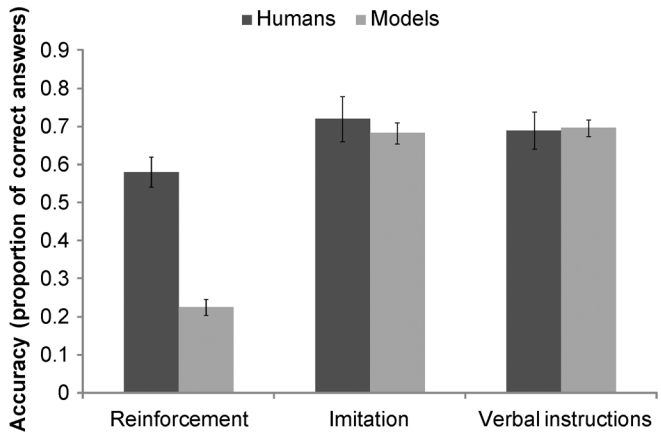


Fig. 8. Accuracy of human participants and models, with standard error bars.

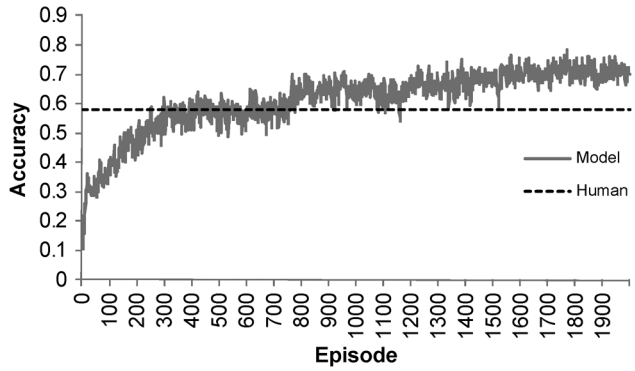


Fig. 9. Accuracy as a function of training episodes in the SARSA-based model of reinforcement learning (learning rate of 0.1 and discount factor of 1.0).

III. Results

We measured models' accuracy, selection complexity and asymmetry, training effort, and network size. Training effort was measured in epochs, and network size was measured in number of recruited hidden units. To match variances of model and human samples, we used the same number of simulations as there were human participants per experimental condition, that is, 21.

A. Comparison of Model and Human

1) Accuracy: First, we compared the accuracy pattern of humans and models, illustrated in Fig. 8. Number of learning episodes of the reinforcement learning models was matched with overt human learning (18.6 trials or episodes). An epoch is a pass through the 24 possible cases, one per episode. Thus humans had less than one epoch of training. Prior to analyses, we performed arcsine data transformations on accuracies to stabilize variance [39].

A one-way ANOVA with model type as an independent factor (three levels: reinforcement, imitation and instruction) revealed a main effect, $F(2, 60) = 104$, $p < 0.001$. Tukey HSD post-hoc tests revealed that the imitation and instructions models formed a homogeneous subset of higher accuracy than the reinforcement model, a pattern of result that is consistent with human performance [13]. We confirmed the hypothesis that imitation and instruction groups outperformed the reinforcement learning group using a planned comparison with contrast coefficients of -2 1 1 for the reinforcement, imitation and instruction groups respectively. The contrast was significant for both humans, $t(60) = 2.4$, $\mathrm{p} < 0.05$, and models, $t(60) = 14.5$, $\mathrm{p} < 0.001$. In other words, for both models and humans, learning by imitation and instructions yielded higher accuracy than learning by environmental rewards alone. This reflects the richness of information available to problem solvers for learning. Whereas problem solvers trained by reinforcement only receive a single binary piece of information per episode after the third weighing, problem solvers trained by imitation and instructions get fully specified targets on what to do for every weighing.

Models trained by reinforcement were less accurate than humans given equivalent training. In fact, models needed about 500 epochs to reach human-level performance, as we see on Fig. 9. This graph also shows that models learned well, increasing in accuracy with training. As we will argue in the discussion, this suggests humans likely engage other cognitive processes, such as reasoning, when elaborating solutions.

2) Selection Complexity and Asymmetry: To compare models and humans at a more precise level, we computed

[4]We used Softmax in training to allow networks some exploration, but we used Hardmax in testing to exploit the solution that the model currently considers the best. A similar behavior may be expected of humans: when tested, they would do their best (i.e., pick actions with highest expected reward), but they would explore more alternatives when learning.

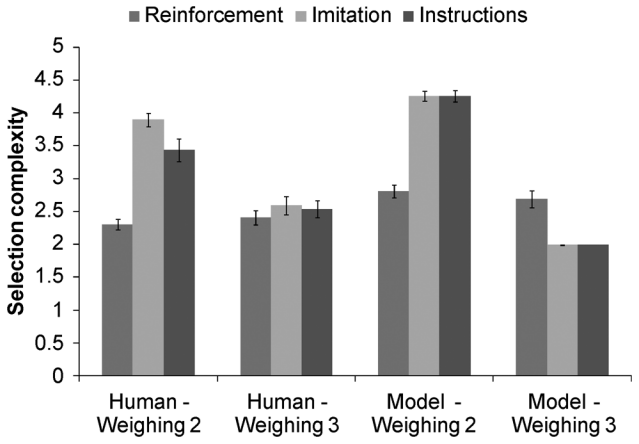


Fig. 10. Complexity of human and model gizmo selections for the second and third weighings, with standard error bars.

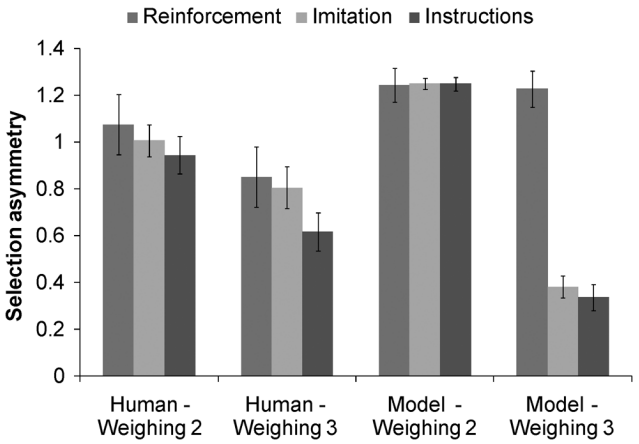


Fig. 11. Asymmetry of human and model gizmo selections for the second and third weighing with standard error bars.

complexity and asymmetry of gizmo selections. We ignored the first weighing because all gizmos were labelled as Unknown in the initial state which yielded a fixed and predictable selection complexity of 2 (Labels U on each side) and an asymmetry of 0.

Fig. 10 and Fig. 11 present results of gizmo selection complexity and asymmetry.

As we can see, demonstrations and instructions effectively reduced human's simplicity bias at the second weighing, as argued in [22]. More generally, we see that availability of demonstrations (imitation) and instructions appeared to result in solution steps that were more complex and asymmetrical in the second weighing than the third weighing, whereas there was no such drop in the reinforcement learning group.

We tested these hypotheses using planned comparisons with contrast coefficients $-1\ 2\ 2$, $-1\ -1\ -1$ applied to the complexity measure and $1\ 1\ 1$, $1\ -2\ -2$ to the asymmetry measure, for human and model solutions. All contrast analyses were significant—model complexity, $F(2,60) = 2537$, $p < 0.001$; human complexity, $F(2,60) = 1304$, $p < 0.001$; model asymmetry, $F(2,60) = 738$, $p < 0.001$; and human asymmetry, $F(2,60) = 240$, $p < 0.001$.

TABLE III
TRAINING PERFORMANCE COMPARISON OF THE IMITATION AND INSTRUCTIONS MODELS

	Epochs	Recruits
Imitation model	418.7	3.38
Instruction model	346.4	2.71

B. Comparison of Imitation and Instruction Models

Table III summarizes training effort, as measured by the number of training epochs, and network sizes for the imitation and instruction models. To improve normality, we compared log-transformed measures of size and training effort. Although the instruction model appeared to train slightly faster, the difference was not significant under the current sample size, $t(41) = 1.7$, $p > 0.05$. Similarly, network sizes did not differ although there was a trend towards significance, $t(41) = 1.9$, $p > 0.05$. In short, we could not reject the null hypotheses stating that training efforts and network sizes were the same irrespectively of the availability of selection rules. However, rules (selected 23 times) were preferred to complexity-matched controls (selected 5 times), suggesting they were nevertheless more useful to solve the task than complexity-matched controls.

IV. DISCUSSION

To summarize, our models collectively captured the pattern of human accuracy in the gizmo task: problem solvers in imitation and instructions learning conditions were more accurate than those in a reinforcement learning condition, as we can see in Fig. 8. Models also captured the pattern of complexity and asymmetry of human selection actions, namely that the imitation and instruction groups exhibited larger complexity and asymmetry than the reinforcement group in the second weighing compared to the third.

We also found in knowledge-based neural learning that, when available, selection rules were preferred as recruits over complexity-matched controls. However, these selection rules had little effect on learning efficiency. Our models suggest that instructions can be represented as pre-trained neural networks and included as modules capable of being recruited. Rule networks may be recruited, that is, used in a final solution, if they correlate well with residual network error.

A. Interpretation of Complexity and Asymmetry Results

The patterns of complexity and asymmetry found reflect the fact that problem solvers in the imitation and instructions groups generated better solutions than those in the reinforcement group. As illustrated in Fig. 2, optimal solutions involved complex arrangements on the second weighing to maximize the information extracted. When done properly, there were at most three possibilities left for the third weighing; leaving little room for complex arrangements. In contrast, problem solvers who extracted less information from their second weighing were left with more possibilities on their third weighing and thus more potential for complex arrangements.

B. Connectionist Versus Symbolic Models of Problem Solving

These connectionist models represent valuable alternatives to traditional symbolic models of problem solving, for at least three important reasons. First, they are plausible—processing in connectionist models and brains is distributed across highly interconnected networks of nonlinear processing units (or neurons). Second, connectionist models generalize to unseen or unvisited problem states. Third, connectionist models are more autonomous, as there is no need for the designer to create or fine tune rules. Neural networks learn them from examples and experience.

C. Relevance to Developmental Robotics

Our CC networks, in addition to being cognitive models, could be used in artificial systems to solve problems. Learning and development are critical to autonomous robots, in terms of not requiring built in reasoning skills. Like our networks, robots could start with knowledge of how to solve difficult problems and learn how to do it through reinforcement, imitation, or instruction, particularly the latter two [40].

D. Our Model in Reference to Other Techniques

Our work relates to that of Fu and Anderson [7], one of the very few models of human problem solving concerned with learning. The authors describe it as a model of rapid, nondeliberative decision making. In contrast, we modeled a more complex and planning-intensive task. The tasks investigated by Fu and Anderson required only a few production rules to choose actions. This contrasts with the large number of possible actions in our task, often on the order of hundreds from a given state. The number of modeler-defined rules or productions may increase as problem difficulty and complexity augment. Because such productions need to be stored explicitly, production systems may have difficulty scaling up to difficult tasks, a problem they share with lookup tables. In contrast, neural network function approximators only require a specification of the input and output space. The "rules" that map the inputs (combination of state and actions) onto the outputs (value of taking action from state) are implicit in the nonlinear mapping function learned. These mappings generalize to unseen or unvisited conditions based on similarity. Generating responses to new states as a graded function of their resemblance to known states is psychologically plausible, and follows more naturally from the processing of connectionist systems than rule-based or production systems.

E. Notes About the Reinforcement Learning Model

Although our models exhibited a pattern of accuracy similar to humans, models of the reinforcement learning condition ($\mathrm{M} = 0.22$) were less accurate than human participants ($\mathrm{M} = 0.58$). We should stress that this discrepancy does not show a failure of the reinforcement learning system. As illustrated in Fig. 9, our SARSA-based system does indeed learn the task, improving in accuracy over training episodes. Fu and Anderson have reported training times on the order of hundreds of trials [7]. The training times we found for the gizmo task, a much more complex problem in terms of search space size, thus appear reasonable.

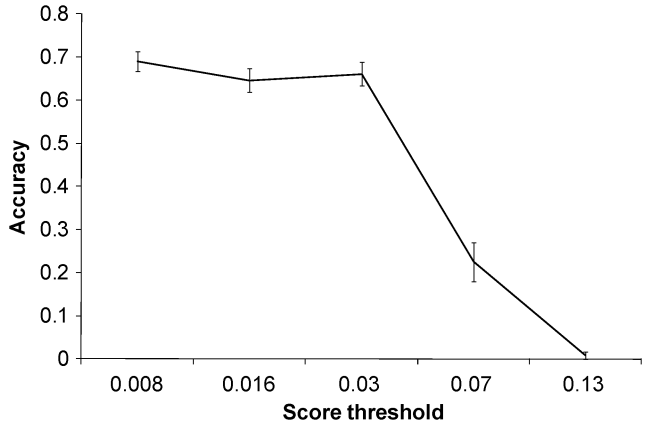


Fig. 12. Mean accuracy of 20 imitation learning models trained with five problem demonstrations as a function of score threshold. Error bars represent SE.

One needs to appreciate how little overt opportunity humans had for learning this problem (less than 20 episodes or trials; that is, less than a single epoch). In future research, a number of hypotheses could be explored to explain this discrepancy. For instance, humans may well engage additional cognitive processes when elaborating solutions to gizmo problems to speed up learning, in addition to some form of temporal-difference mechanism.

1) Reasoning or mental rehearsing. Models need to overtly perform actions to obtain rewards and learn. In contrast, humans can mentally play different alternatives and look ahead. For instance, mental search over a certain number of plies is well-documented in chess playing [41], [42]. A similar mechanism may allow participants to predict rewards they would get without needing to take those actions. To model reasoning in a future model improvement, we could use techniques such as TD-leaf which combines search with reinforcement learning in a single system [43].
2) Means-ends analysis. In combination with reasoning, means-ends analysis [9] can allow humans to select actions that move them closer to goal states. Humans might get relatively rich information due to self-generated rewards based on distance to goals.

V. Conclusion

This work introduced learning-centric modeling techniques to the study of problem solving. Problem solving has seldom been studied from such a learning-centric perspective. For instance, aside from [7] and [8], very few, if any, cognitive models of complex problem solving with SARSA-based reinforcement learning exist.

We have seen that models capture the accuracy, complexity and asymmetry patterns of human solutions. This work has also shown that concepts and representations used in information processing theory (states, actions, and search spaces) can be successfully used in a connectionist context for three different kinds of learning models: imitation, instruction and reinforcement. Because they all use connectionist function approximators, our models learn readily, generalize well and are neurologically plausible. KBCC neural networks are particularly pow-

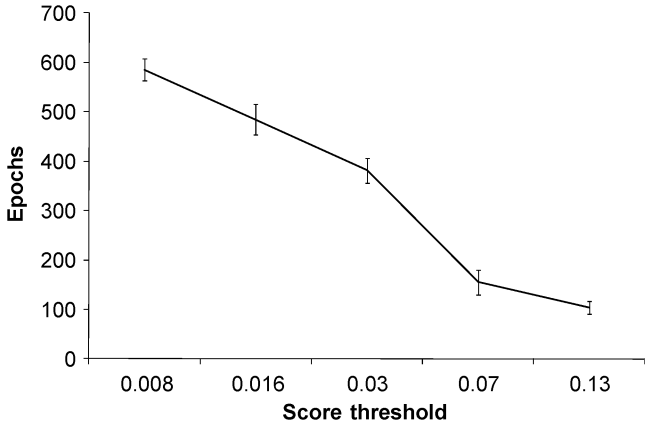


Fig. 13. Mean number of epochs to train 20 imitation learning models with five problem demonstrations as a function of score threshold. Error bars represent SE.

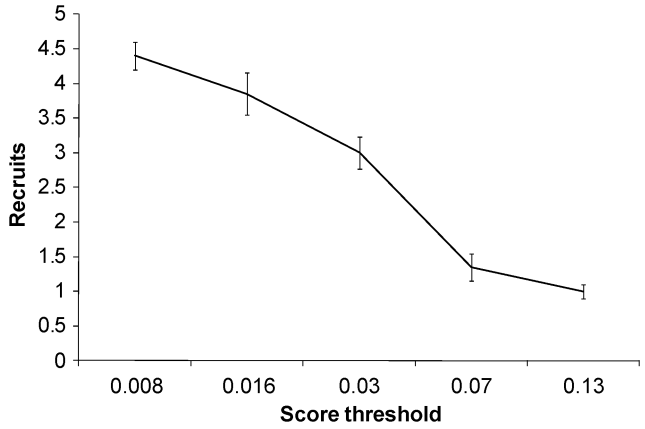


Fig. 14. Mean number of cascade-correlation recruits after training of 20 imitation learning models with five problem demonstrations as a function of cascade-correlation score threshold. Error bars represent SE.

erful learners that grow as they learn and incorporate other, previously trained networks.

These models were concerned with the modeling of learning in human adults. In future research, the developmental aspect of learning to solve problems could be investigated.

Additional follow up work could explore how to account for reasoning and means-ends analysis in these learning-centric models. These additions would likely improve coverage of human problem solving and may help us to better understand a hallmark of intelligence: how humans learn to solve difficult problems.

Appendix
Effect of Cascade-Correlation Score Threshold in the Imitation Learning Model

We investigated the effect of CC score-threshold parameter on the training of imitation learning models. We measured accuracy, training epochs and recruits with score threshold values equal to 0.1, 0.2, 0.4, 0.8, and 1.6 times the interval between proportions.[5] Twenty networks were trained per condition, for a total of 100 networks. Mean accuracy (proportion of correct answers) on the 24 test problems is shown in Fig. 12, number of SDCC training epochs in Fig. 13, and number of SDCC recruits in Fig. 14.

A one-way ANOVA revealed an effect of score threshold on accuracy, $\mathrm{F}(4,99) = 115$, $\mathrm{p} < 0.001$. Tukey HSD post-hoc tests further revealed that accuracy was stable over a range of low score thresholds (0.008, 0.016, and 0.03) and drops significantly for score thresholds above 0.03. Furthermore, score threshold had a significant effect on the number of training epochs, $\mathrm{F}(4,99) = 74$, $\mathrm{p} < 0.001$, and on the number of recruited units, $\mathrm{F}(4,99) = 49$, $\mathrm{p} < 0.001$. As we see in the graphs, lower score thresholds, which result in deeper training, increased the number of training epochs and recruitments necessary to learn the task. Score threshold values of $0.4 * 1/12 = 0.033$ minimize training time and number of recruitments while maintaining a high accuracy. Therefore, we used that value in simulations of the imitation and instruction models.

[5]The [0,1] output interval codes proportions for 12 gizmos, thus is divided in 12, yielding score threshold values of about 0.008, 0.016, 0.3, 0.7, and 0.13.

Acknowledgment

The authors would like to thank François Rivest for his insightful comments and suggestions. They also thank Kristine H. Onishi for feedback on an early version of the manuscript.

References

[1] J. H. Holland, K. J. Holyoak, R. E. Nisbett, and P. Thagard, *Induction—Processes of Inference, Learning and Discovery*. Cambridge, MA: MIT Press, 1986.

[2] K. J. Holyoak and P. Thagard, *Mental Leaps—Analogy in Creative Thought*. Cambridge, MA: MIT Press, 1996.

[3] A. Newell, *Unified Theories of Cognition*. Cambridge, MA: Harvard University Press, 1990.

[4] G. Polya, *How to Solve It*, 2nd ed. Princeton, NJ: Princeton University Press, 1957.

[5] J. R. Anderson, D. Bothell, M. D. Byrne, D. Douglass, C. Lebiere, and Y. Qin, "An integrated theory of mind," *Psychol. Rev.*, vol. 111, pp. 1036–1060, 2004.

[6] G. Gunzelmann and J. R. Anderson, "Problem solving: Increased planning with practice," *Cogn. Syst. Res.*, vol. 4, pp. 57–76, 2003.

[7] W.-T. Fu and J. R. Anderson, "From recurrent choice to skill learning: A reinforcement-learning model," *J. Exper. Psychol.: General*, vol. 135, pp. 184–206, 2006.

[8] S. Nason and J. E. Laird, "Soar-RL: Integrating reinforcement learning with Soar," in *Proc. Sixth Int. Conf. Cogn. Modeling* , Mahwah, NJ, 2004, pp. 208–213, Erlbaum.

[9] A. Newell and H. A. Simon, *Human Problem Solving*. Englewood Cliffs, NJ: Prentice-Hall, 1972.

[10] G. Katona, *Organizing and Memorizing*. New York: Columbia University Press, 1940.

[11] M. Carpenter, J. Call, and M. Tomasello, "Understanding "prior intentions" enables two-year-olds to imitatively learn a complex task," *Child Dev.*, vol. 75, pp. 1431–1441, 2002.

[12] R. H. Cuijpers, H. T. Schie, M. Koppen, W. Erlhagen, and H. Bekkering, "Goals and means in action observation: A computational approach," *Neural Netw.*, vol. 19, pp. 311–322, 2006.

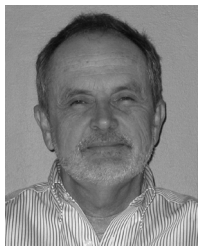
[13] F. Dandurand, M. Bowen, and T. R. Shultz, "Learning by imitation, reinforcement and verbal rules in problem solving tasks," in *Proc. Third Int. Conf. Dev. Learning: Dev. Social Brains*, J. Triesch and T. Jebara, Eds., La Jolla, CA, 2004, pp. 88–95, University of California, San Diego, Institute for Neural Computation.

[14] D. C. Noelle and G. W. Cottrell, "A connectionist model of instruction following," in *Proc. 17th Annu. Conf. Cogn. Sci. Soc.*, J. D. Moore and J. F. Lehman, Eds., Hillsdale, NJ, 1995, pp. 369–374, Lawrence Erlbaum.

[15] G. Biele, J. Rieskamp, and R. Gonzalez, "Computational models for the combination of advice and individual learning," *Cogn. Sci.*, vol. 33, pp. 206–242, 2009.
[16] G. B. Kaplan and C. Güzelis, "Hopfield networks for solving Tower of Hanoi problems," *ARI: An Interdiscipl. J. Phys. Eng. Sci.*, vol. 52, pp. 23–29, 2001.
[17] R. W. Parks and J. Cardoso, "Parallel distributed processing and executive functioning: Tower of Hanoi neural network model in healthy controls and left frontal lobe patients," *Int. J. Neurosci.*, vol. 89, pp. 217–240, 1997.
[18] R. W. Parks, D. S. Levine, and D. L. Long, *Fundamentals of Neural Network Modeling: Neuropsychology and Cognitive Neuroscience*. Cambridge, MA: MIT Press, 1998, p. 428.
[19] H. A. Simon, "The structure of ill-structured problems," *Artif. Intell.*, vol. 4, pp. 181–202, 1973.
[20] M. L. Simmel, "The coin problem: A study in thinking," *Amer. J. Psychol.*, vol. 66, pp. 229–241, 1953.
[21] L. Halbeisen and N. Hungerbuhler, "The general counterfeit coin problem," *Discr. Math.*, vol. 147, pp. 139–150, 1995.
[22] F. Dandurand, T. R. Shultz, and K. H. Onishi, "Strategies, heuristics and biases in complex problem solving," in *Proc. Twenty-Ninth Meet. Cogn. Sci. Soc. (CogSci 2007)*, New York, 2007, pp. 917–922, Lawrence Erlbaum Associates.
[23] N. Chater and P. Vitányi, "Simplicity: A unifying principle in cognitive science?," *Trends Cogn. Sci.*, vol. 7, pp. 19–22, 2003.
[24] J. Freyd and B. Tversky, "Force of symmetry in form perception," *Amer. J. Psychol.*, vol. 97, pp. 109–126, 1984.
[25] S. E. Fahlman and C. Lebiere, "The cascade-correlation learning architecture," in *Advances in Neural Information Processing Systems 2*. Los Altos, CA: Morgan Kaufmann, 1990, pp. 524–532.
[26] T. R. Shultz, *Computational Developmental Psychology*. Cambridge, MA: MIT Press, 2003.
[27] S. Baluja and S. E. Fahlman, Reducing Network Depth in the Cascade-Correlation Carnegie Mellon University. Pittsburgh, PA, CMU-CS-94-209, 1994.
[28] F. Dandurand, V. Berthiaume, and T. R. Shultz, "A systematic comparison of flat and standard Cascade-Correlation using a student-teacher network approximation task," *Connect. Sci.*, vol. 19, pp. 223–244, 2007.
[29] S. E. Fahlman, "Faster-learning variations on back-propagation: An empirical study," in *Proc. 1988 Connect. Models Summer School*, T. J. Sejnowski, G. E. Hinton, and D. S. Touretzky, Eds., San Mateo, CA, 1988, Morgan Kaufmann.
[30] T. R. Shultz and F. Rivest, "Knowledge-based cascade-correlation: Using knowledge to speed learning," *Connect. Sci.*, vol. 13, pp. 43–72, 2001.
[31] J. P. Thivierge, F. Dandurand, and T. R. Shultz, "Transferring domain rules in a constructive network: Introducing RBCC," in *Proc. 2004 IEEE Int. Joint Conf. Neural Netw. (IJCNN '04)*, 2004, vol. 2, pp. 1403–1408, IEEE.
[32] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.
[33] F. Dandurand, T. R. Shultz, and F. Rivest, "Complex problem solving with reinforcement learning," in *Proc. Sixth IEEE Int. Conf. Dev. Learning (ICDL-2007)*, 2007, pp. 157–162, IEEE.
[34] R. E. Suri and W. Schultz, "A neural network model with dopamine-like reinforcement signal that learns a spatial delayed response task," *Neuroscience*, vol. 91, pp. 871–890, 1999.
[35] J. C. Houk, J. L. Adams, and A. G. Barto, "A model of how the basal ganglia generate and use neural signals that predict reinforcement," in *Models of Information Processing in the Basal Ganglia*, J. C. Houk, J. L. Davis, and D. G. Beiser, Eds. Cambridge, MA: MIT Press, 1995, pp. 249–270.
[36] K. Samejima, Y. Ueda, K. Doya, and M. Kimura, "Representation of action-specific reward values in the striatum," *Science*, vol. 310, pp. 1337–1340, 2005.
[37] N. Cowan, "The magical number 4 in short-term memory: A reconsideration of mental storage capacity," *Behav. Brain Sci.*, vol. 24, pp. 87–125, 2000.
[38] G. A. Miller, "The magical number seven, plus or minus two: Some limits on our capacity for processing information," *Psychologic. Rev.*, vol. 63, pp. 81–97, 1956.
[39] R. V. Hogg and A. T. Craig, *Introduction to Mathematical Statistics*. Upper Saddle River, NJ: Prentice-Hall, 1995.
[40] T. R. Shultz, F. Rivest, L. Egri, J.-P. Thivierge, and F. Dandurand, "Could knowledge-based neural learning be useful in developmental robotics? The case of KBCC," *Int. J. Human. Robot.*, vol. 4, pp. 245–279, 2007.
[41] D. H. Holding, *The Psychology of Chess Skill*. New York: Lawrence Erlbaum , 1985.
[42] D. H. Holding, "Theories of chess skill," *Psychologic. Res.*, vol. 54, pp. 10–16, 1992.
[43] J. Baxter, A. Tridgell, and L. Weaver, "TDLeaf(lambda): Combining temporal difference learning with game-tree search," in *Proc. Ninth Australian Conf. Neural Netw.*, 1998, pp. 168–172.

Frédéric Dandurand (M'08) completed the Ph.D. degree in psychology at McGill University, Canada, in 2007.

He is currently a Postdoctoral Researcher at the French Centre National de la Recherche Scientifique (CNRS). His principal research interest is the computational modeling of high-level cognitive processes such as language and problem solving. He is also a Professional Engineer in Canada and has worked as a software engineer for two major computer equipment companies.



Thomas R. Shultz received the Ph.D. degree in psychology from Yale University, New Haven, CT.

He is Professor of Psychology and Associate Member of the School of Computer Science at McGill University, Canada. His current research interests include connectionism, cognitive science, cognitive development, cognitive consistency phenomena, constraint satisfaction, relations between knowledge, and learning, and evolution.

Dr. Shultz is a Member of the IEEE Neural Networks Society Autonomous Mental Development Technical Committee and Chair of the IEEE Autonomous Mental Development Task Force on Developmental Psychology.