International Journal of Humanoid Robotics
Vol. 4, No. 2 (2007) 245–279

World Scientific
www.worldscientific.com

COULD KNOWLEDGE-BASED NEURAL LEARNING BE USEFUL IN DEVELOPMENTAL ROBOTICS? THE CASE OF KBCC

THOMAS R. SHULTZ[*,†,¶], FRANÇOIS RIVEST[‡,‖], LÁSZLÓ EGRI[†,**], JEAN-PHILIPPE THIVIERGE[§,††] and FRÉDÉRIC DANDURAND[*,‡‡]

[*]*Department of Psychology, McGill University, 1205 Penfield Avenue, Montreal, QC H3A 1B1, Canada*

[†]*School of Computer Science, McGill University, 3480 University Street, Montreal, QC H3A 2B4, Canada*

[‡]*Département d'Informatique et de Recherche Opérationnelle, Université de Montréal, CP 6128 succursale Centre Ville, Montréal, QC H3C 3J7, Canada*

[§]*Département de Physiologie, Université de Montréal, CP 6128 succursale Centre Ville, Montréal, QC H3T 1J4, Canada*

[¶]*thomas.shultz@mcgill.ca*
[‖]*francois.rivest@mail.mcgill.ca*
[**]*laszlo.egri@mail.mcgill.ca*
[††]*jean-philippe.thivierge@umontreal.ca*
[‡‡]*frederic.dandurand@gmail.com*

Received 30 August 2006
Revised 10 March 2007

The new field of developmental robotics faces the formidable challenge of implementing effective learning mechanisms in complex, dynamic environments. We make a case that knowledge-based learning algorithms might help to meet this challenge. A constructive neural learning algorithm, knowledge-based cascade-correlation (KBCC), autonomously recruits previously-learned networks in addition to the single hidden units recruited by ordinary cascade-correlation. This enables learning by analogy when adequate prior knowledge is available, learning by induction from examples when there is no relevant prior knowledge, and various combinations of analogy and induction. A review of experiments with KBCC indicates that recruitment of relevant existing knowledge typically speeds learning and sometimes enables learning of otherwise impossible problems. Some additional domains of interest to developmental robotics are identified in which knowledge-based learning seems essential. The characteristics of KBCC in relation to other knowledge-based neural learners and analogical reasoning are summarized as is the neurological basis for learning from knowledge. Current limitations of this approach and directions for future work are discussed.

Keywords: Knowledge-based learning; neural networks; knowledge transfer; developmental robotics.

1. Introduction

Solutions to some of the most difficult problems in robotics are being increasingly inspired by studies of human cognition and development,[1] forming one of the key threads of the new field of developmental robotics, at the interface between robotics and developmental psychology. One of most promising avenues of exploration is the design of humanoid robots able to autonomously learn and develop in interaction with natural environments containing other robots or humans. A variety of learning algorithms, including reinforcement and connectionist approaches, have been employed successfully in this context. However, these algorithms typically lack a crucial property of human cognition: the ability to recruit and re-use existing knowledge when learning a novel task. Therefore, we expect that enabling the use of existing knowledge could significantly improve the performance of robot learning algorithms. Adaptive algorithms that take advantage of knowledge would also make a better match to human learning and development. Because of these considerations, robot designs that incorporate autonomous developmental processes are starting to employ knowledge transfer as a central component of learning and development.[2,3]

In this article, we discuss the prospects of knowledge-based learning in the context of knowledge-based cascade-correlation (KBCC), a neural learning algorithm that we have been developing in recent years.[4] The general idea behind knowledge transfer in KBCC is to build new learning on top of existing knowledge rather than starting from scratch on each new learning task. This property of KBCC creates the possibility for incremental, lifelong learning, wherein knowledge is acquired, stored, retrieved, and reused over extended periods of time, during which a learner (e.g. a robotic agent) may be exposed to many different tasks across a variety of contexts and environments. By discovering similarities between previous and novel tasks and contexts, the learner could conceivably make effective use of prior knowledge to improve its performance. Although other computational approaches, such as Bayesian learning,[5] also provide a natural way to incorporate knowledge into learning, we focus here on learning in artificial neural networks because of their mechanistic nature, apparent biological realism, and demonstrated ability to simulate many aspects of human development and learning in a principled fashion.[6,7]

Although neural networks are among the most successful approaches to modeling learning and development,[6] a significant limitation of such neural learning is that it is typically conducted *from scratch*, without allowing for the influence of existing knowledge. For simulation purposes, this is implausible because people make extensive use of their existing knowledge when learning.[8–14] Use of knowledge is largely responsible for the ease and speed with which people are able to learn, as well as for interference of learning by existing knowledge. Typical neural networks fail to use knowledge while learning because they start learning from random connection weights. From a computational point of view, when the hypothesis-space is constrained, not only by the examples, but also by existing knowledge, that space becomes easier to search. Thus, using knowledge can speed learning and require fewer training examples.

In this article, we examine a neural-learning algorithm that uses its knowledge to learn new problems. This algorithm is an extension of cascade-correlation (CC), a constructive learning algorithm that has proved useful in simulating more than a dozen phenomena in learning and cognitive development.[7] Ordinary CC creates a network topology by recruiting new hidden units into a feed-forward network as needed in order to reduce error.[15] The algorithm extension, called knowledge-based cascade-correlation (KBCC), recruits whole previously-learned networks in addition to the single hidden units recruited by CC.[4]

KBCC is similar in spirit to recent neural-network research on inductive transfer,[16] multitask learning,[17] lifelong learning,[18] sequential learning,[19] knowledge insertion,[20] modularity,[21] and input re-coding.[22] We show that KBCC incorporates and integrates many of these ideas by learning, storing, finding, mapping, and recruiting knowledge within a constructive-learning approach. In doing so, it often outperforms other knowledge-based learners.

We describe the KBCC algorithm, review its performance on toy problems, realistic engineering problems, and psychology simulations. We then identify some additional domains likely to be of interest to developmental robotics in which knowledge-based learning seems important. We also summarize evidence on the neurological basis for learning from knowledge, discuss the advantages and current limitations of KBCC, and highlight some promising directions for future work that seems relevant to robotics. Comparison to other methods, particularly knowledge-based neural learning algorithms and analogical and case-based reasoning, is made where appropriate.

2. KBCC

As noted, KBCC is a variant of CC, a feed-forward constructive algorithm that grows a network while learning, essentially by recruiting new hidden units as needed.[15] The principal innovation in KBCC is the potential to recruit previously-learned networks or indeed any differentiable function, in competition with single hidden units. The computational device that gets recruited is the one whose output covaries best with residual network error, just as in ordinary CC. As in standard feed-forward backpropagation networks, the output of a unit is given by a weighted sum from the feeding units processed through an activation function, which can be a sigmoid. Hidden nodes are processed similarly, but can also be whole networks in the case of KBCC.

At the start of learning, a KBCC network (see Fig. 1 for an example) has only a bias unit and input units fully connected to output units with randomized weights, which are uniformly distributed in the range $[-1, 1]$. During the initial learning phase, known as the *output phase*, connection weights feeding the output units are trained to minimize the sum of squared error, defined as:

$$F = \sum_{o} \sum_{p} (y_{o,p} - t_{o,p})^2, \tag{1}$$

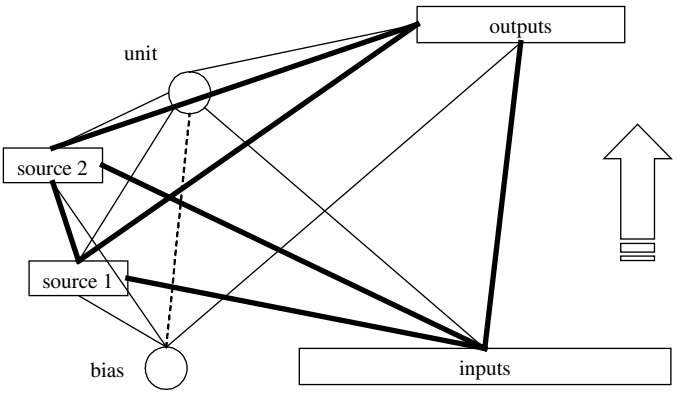


Fig. 1. Hypothetical KBCC network that has recruited two source networks and a sigmoid unit. The dashed line represents a single connection weight, thin solid lines represent weight vectors, and thick solid lines represent weight matrices. The arrow indicates direction of activation flow.

where $y_{o,p}$ is the activation of output o in response to training pattern p, and $t_{o,p}$ is the corresponding target activation value that the network is learning to produce. The gradient with respect to the weights is

$$\frac{\partial F}{\partial w_{u,o}} = \sum_p \left(y_{o,p} - t_{o,p}\right) f' \left(\sum x_{u,p} w_{u,o}\right), \tag{2}$$

where $x_{u,p}$ is the activation of unit u to pattern p, $w_{u,o}$ is the weight between u and o, and f' is the derivative of the activation function of the output node.

The optimization is usually done with Quickprop,[23] a gradient descent algorithm using a linear estimation of the second derivative, with learning rate $\varepsilon = 0.175/n$, maximum growth factor $\mu = 2.0$, weight decay $\gamma = 0.0002$, and n = number of training patterns.

If error reduction stagnates or a certain number of epochs (default of 100) is reached without learning success, then the algorithm shifts to the so-called *input phase*. Stagnation is defined by error F not decreasing by more than a certain amount $\Delta_{F,l}$ over the last few epochs l. By default, $\Delta_{F,l} > -0.01$, and $l = 8$. In input phase, small random weights, uniformly distributed in the range $[-1, 1]$ connect every non-output unit of the target network to each input of each candidate recruit. Those weights are then trained to maximize a covariance between each candidate's outputs c_o and residual error in the target network. The covariance for each candidate is defined as:

$$G_c = \frac{\sum_{c_o} \sum_o Cov\left(Y_c, E\right)^2_{c_o,o}}{O_c O F}, \tag{3}$$

where Y_c is the matrix of candidate c output activations for all patterns, E is the matrix of target network errors for all patterns, $Cov(Y_c, E)$ is the covariance matrix

relating output activations and network errors, O_c is the number of outputs from candidate c, O the number of outputs in the target network, and F is network error as defined by Eq. (1). The gradient of Eq. (3) can also be derived as long as the gradient of each candidate network's output can be computed for the weighted sum values used as their inputs.

Composition of the pool of candidates is what varies the most from one experiment to another. The pool usually has a number of sigmoid or asigmoid logistic units initialized randomly and a number of candidate source networks, i.e. networks previously trained on other tasks. Activation of sigmoid units are in the range $[-0.5, 0.5]$; those of asigmoid units in the range $[0, 1]$. Input weights to multiple copies of source networks are usually initialized with random input weights to improve optimization. Of these copies, one is usually connected using the identity matrix with off-diagonal zeros. This ensures rapid recruitment and learning when the source network has exact knowledge of how to solve the target task.

Input-phase training is done with Quickprop with learning rate $\varepsilon = 0.175/nh$, maximum growth factor $\mu = 2$, weight decay $\gamma = 0.0000$, $h =$ number of values feeding the candidates. The gradient of G_c with respect to candidate c input weights is computed by the chain rule through the candidate. Input-phase training continues until a maximum number of epochs is reached (default of 100) or until the increases in covariances stagnate ($G_c > 0.2$, $\Delta_{G,l} < 0.03$, and $l = 8$). When the input phase is finished, the candidate with the highest covariance (G) is installed in the network with new connection weights from the outputs of the recruit to the target network's output units. These new weights are initialized with random values, uniformly distributed in the range $[0, 1]$, with a sign which is the negative of the sign of the covariance.

The algorithm then shifts back to output phase to adjust all target-network output weights in order to use the new recruit effectively. KBCC continues to cycle back and forth between output and input phases until learning is complete, or some maximum number of recruits is exceeded. Learning is complete when the absolute error of each output on each pattern is less than score threshold of $\delta = 0.4$. The maximum number of recruits has a default of 25, which has never been reached in our lab, except in some very complex artificial tasks. In both phases, weights are adjusted with the Quickprop algorithm that uses curvature as well as slope of the error surface.[23] A hypothetical KBCC target network with three recruits is illustrated in Fig. 1.

Given a pool of candidates, optimization of the input-side weights to correlate with residual error allows for an effective use of source knowledge. Useful maps are created from the target network input space to the candidate network input space. The input phase thus serves to find the best map to make a candidate useful in reducing residual error in the target network. Once a candidate is installed, the output weights, as well as any further recruitments, integrate the installed source networks into a final solution.

3. Applications of KBCC

KBCC has been applied to toy and realistic problems, as well as psychology simulations.

3.1. *Toy problems*

Although toy problems may not seem as challenging as realistic problems and simulations, they can play an important role in understanding the behavior and ability of complex algorithms such as KBCC.

3.1.1. *Geometric shapes*

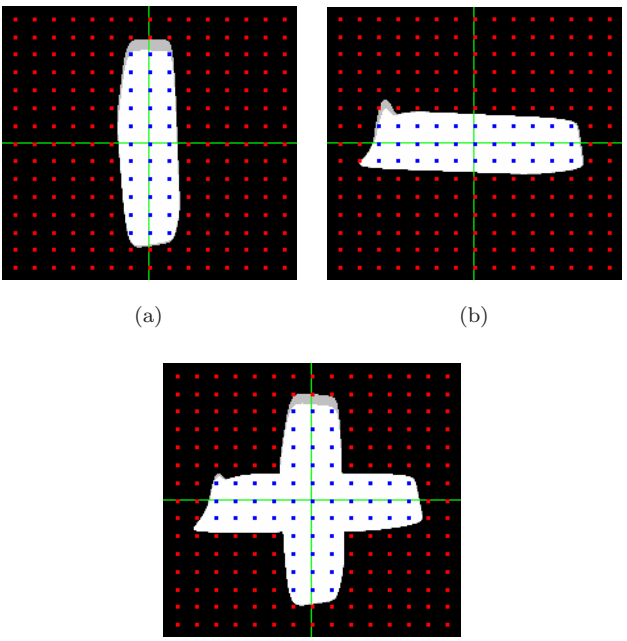
The first batch of toy problems we explored involved learning about two-dimensional geometric shapes under various transformations such as translation, rotation, and size changes, as well as compositions of complex shapes from simpler shapes.[4] Networks had to learn to distinguish points within a target shape from points outside the shape. Autonomous robots could benefit from being able to learn visual shapes and their boundaries.

Plots of activation outputs enabled evaluation of a network's knowledge representations. Candidate recruits included, in different conditions, networks with exact, inexact, or irrelevant knowledge of the target task. Each experiment also contained a control condition with no knowledge at all. Default parameter settings were used throughout these experiments.

An illustration of shape learning with KBCC is presented in Fig. 2. To learn a cross shape, this network recruited previously-learned vertical and horizontal rectangles, greatly shortening learning time, and lessening the number of recruits and connection weights.[4] Notice that the recruited components of the cross in Fig. 2 are very similar to their original sources and that the original sources remain unaltered in the composition. These are characteristics of classical, concatenative compositionality, in which components are incorporated unaltered, something that was claimed to be impossible for neural networks but common in humans.[24,25]

With a single source of knowledge in memory, ANOVAs revealed that KBCC learned fastest with exact knowledge, followed by exact but embedded knowledge, close and relevant knowledge, distant relevant knowledge, irrelevant knowledge, and no knowledge at all. When learning a multi-component target task (see Fig. 2), KBCC learned faster with knowledge of both components than with knowledge of only one component. With multiple sources of knowledge, KBCC showed a tendency to recruit sources in this same order, that is, to recruit the source that allowed it to learn faster.

To quantify the amount of learning speed up in the various experiments, we divided mean epochs to learn in a no-knowledge condition by mean epochs in an exact-knowledge condition. These measures, plotted in Fig. 3, range from 3.7 in the rotation experiment to 15.5 in the components experiment. Learning speed should



(a) (b)

(c)

Fig. 2. Output activation diagrams for a KBCC network that learned a cross target (c) by recruiting its two components (a and b). Dark dots represent training points inside a shape; light dots training points outside a shape. White background indicates generalization to test points inside a shape, black background indicates generalization to test points outside the shape, and gray background indicates intermediate values.

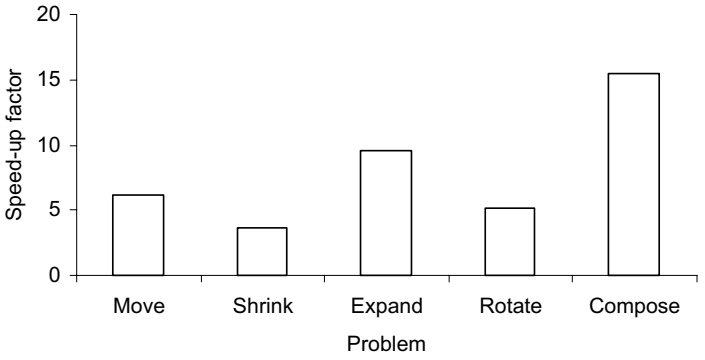


Fig. 3. Speed-up factors in geometric shape experiments plotted as mean epochs in the no-knowledge condition divided by mean epochs in the exact-knowledge condition.

be a prime consideration in robots that have to learn and act in real time. Finally, there was a strong tendency for networks to recruit relevant knowledge whenever it was available.

Generalization was tested with 200 randomly chosen test patterns. The mean proportion of misclassifications by KBCC networks across experiments was 0.03. Generalization did not vary much across these experiments because learning was allowed to continue until all outputs on all patterns were within a particular score threshold of target activations. Some new experiments show that, if learning is terminated sooner, then generalization is better if more relevant knowledge has been recruited. Such incomplete training is likely to be more characteristic of realistic online robotic learning, where generalization is likely to be an appropriate measure of success.

KBCC networks learned translation problems faster than Multitask Learning (MTL) networks did.[26] An MTL network is trained in parallel on several tasks from the same domain, with a single output unit for each task.[17] Such MTL networks typically learn a common hidden-unit representation, which can be useful for subsequent learning of tasks in the same domain. But in this case, ANOVAs showed that MTL networks did not exhibit any benefits of knowledge in terms of increased learning speed over knowledge-free networks. In contrast, ANOVAs showed that KBCC learning was fastest with exact knowledge followed, in turn, by inexact but relevant knowledge (e.g. a rectangle in a different location), less relevant knowledge (e.g. a circle), and no knowledge at all.

MTL networks had particular difficulty extracting exact knowledge from an overly complex source network. Moreover, they often failed to learn their assigned source problem and thus had to be replaced before proceeding to the target phase. In contrast, KBCC networks always learned their source problems successfully. Analysis showed that the primary reason that MTL did not speed the learning of new tasks is that it required both old and new tasks to be freshly learned in parallel. In contrast to this, KBCC recruited its old knowledge without having to relearn it.

3.1.2. *Parity*

Other toy, but difficult problems involved learning high-dimensional parity problems with knowledge of smaller parity problems.[26] Parity problems require a network to turn on an output unit only when an odd number of binary inputs are on. Generalization in such problems has been notoriously difficult to demonstrate. High-dimensional parity problems are probably not likely to be encountered by autonomous robots, but they are examples of problems that are difficult to learn and resistant to generalization, characteristics common to natural environments.

We designed an experiment with a pool of 12 candidates. Across four conditions, the contents of the pool varied as follows: (a) 12 sigmoid units, (b) six sigmoids and six parity-2 networks, (c) six sigmoids and six parity-4 networks, and (d) four of each type of candidate. In each of these four conditions, 30 KBCC target networks were trained on a parity-8 problem with eight binary input units.

KBCC networks learned parity-8 problems significantly faster and with fewer recruits ($p < 0.05$) when parity-4 networks were included in the KBCC candidate source pool [conditions (c) and (d)]. Parity-4 networks tended to be recruited by KBCC target networks whenever available. In contrast, parity-2 networks were avoided because CC can learn a parity-2 problem by recruiting a single sigmoid unit.

3.1.3. *Chessboard shapes*

Similarly, KBCC networks learned complex chessboard shapes from knowledge of simpler chessboards.[26] Chessboard shapes could be representative of visual stimuli that a robot might need to learn about. As with parity, networks here used simpler previous knowledge (2×2 and 4×4 chessboards) to compose a solution to a similar, but more complex 8×8 problem. Content of the candidate pool was varied to study the effect of prior knowledge on new learning. In four different conditions, the pool contained (a) 12 sigmoid units, (b) six sigmoids and six 2×2 networks, (c) six sigmoids and six 4×4 networks, or (d) four of each type of candidate. All types of candidates were recruited, but 4×4 source networks were preferred whenever available.

Mean recruits and mean epochs to learn the 8×8 chessboard shape are presented in a double-y plot in Fig. 4 for the four different candidate-pool conditions. ANOVA and follow-up multiple comparison of means revealed that target networks in conditions with 4×4 source networks in the pool [(c) and (d)] required fewer recruits and learned faster than did target networks in the other two conditions [(a) and (b)], $p < 0.05$. Furthermore, target networks in condition (b) (with some 2×2 source networks) learned faster and with fewer recruits than did target networks in condition (a) (with sigmoid candidates only).

Target-network solutions can be visualized by plotting output activations after training. For example, Fig. 5 shows output activations and training patterns of a KBCC network having learned an 8×8 chessboard shape after recruiting a 4×4

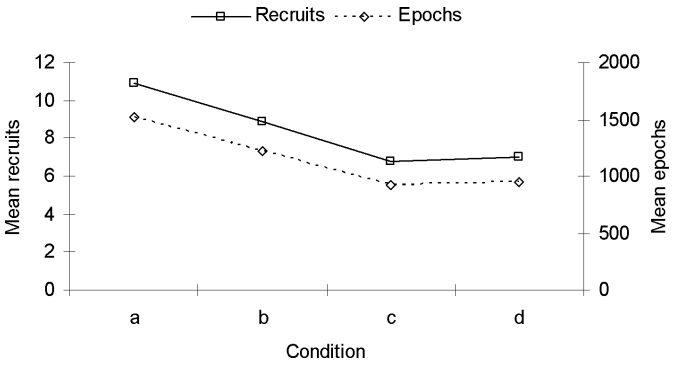


Fig. 4. Mean recruits and mean epochs to learn an 8×8 chessboard shape in four different candidate-pool conditions.

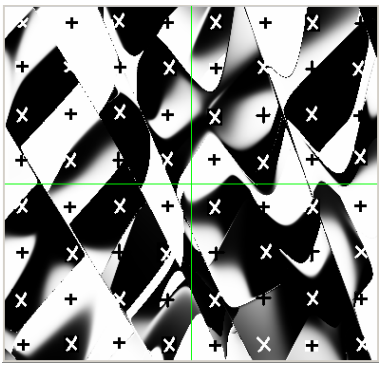


Fig. 5. Output activation plot for a KBCC network that learned an 8×8 chessboard shape. The training patterns are marked with + for positive output and × for negative output. Generalization is indicated by background color: white for positive responses, black for negative responses, and grey for intermediate responses.

chessboard source network. If a target network was correct on training patterns and generalized correctly to nearby test patterns, positive outputs (+) should appear on a white background and negative outputs (×) on a black background. Grey regions are those for which the network gave intermediate, uncommitted answers.

3.2. *Realistic engineering problems*

KBCC has also been applied to some realistic problems, including recognition of spoken vowels and determination of gene splice junctions.

3.2.1. *Vowel recognition*

KBCC has been applied to the Peterson–Barney vowel recognition problem from the CMU AI repository.[27] The dataset contains the two middle formants of the speech sounds of ten different vowels made by 76 speakers: 33 adult males, 28 adult females, and 15 children. The task was to recognize the input vowel by turning on the binary output corresponding to the vowel and turning off the nine other output units. Learning to recognize vowels, and generalizing across speakers, would be essential for robots learning to comprehend human speech. In six different transfer scenarios, CC networks were trained on one type of speaker (e.g. adult males) and then, with those networks as sources, KBCC networks were trained on speakers of a different type (e.g. adult females, or children), creating six different experimental conditions. In three control conditions, CC networks learned the vowels of one of three types of speaker from scratch. Training and testing was done using a ten-fold cross-validation technique.

Peak generalization to untrained vowels was 89% for both CC and KBCC networks. But KBCC networks able to recruit a source network from a different type of speaker learned faster (mean of 827 epochs) than CC networks having to learn

from scratch (mean of 1,279 epochs), $p < 0.005$. As noted previously, learning speed is important for robots having to function under real time constraints.

3.2.2. *Splice junctions*

KBCC and other algorithms were applied to gene splice-junction determination, a benchmark problem from the UCI Machine Learning Repository. The dataset consisted of 1,000 examples each representing a DNA sequence, encoded with 240 binary features. Category labels separated these examples into three groups according to actual splice sites within the sequence. Although we cannot currently envision a need for robots to learn splice junctions, they might well encounter similarly complex categorization problems. Splice-junction detection is, in any case, a difficult and practically important problem.

To prepare KBCC, five separate source networks were trained to identify sequences that satisfied one of five symbolic rules included with the dataset for locating splice junctions. Source networks were trained on sequences where a given rule was satisfied and other sequences where the rule was violated, with target outputs of 1 and 0, respectively. Once trained in this fashion, the rule-based source networks were placed in the candidate pool of a KBCC target network being trained on the main task of identifying splice junctions. KBCC therefore had the option of recruiting these rules that had been learned by source networks.

Because not all source networks were recruited during training, KBCC was useful in identifying the knowledge that was most relevant to success. Of the five rules available, only rules 1 and 2 were recruited. On this task, KBCC required fewer recruits than did knowledge-free CC networks, but did not outperform CC on learning speed and accuracy, probably due to the difficulty of encoding some of the rules in a neural format.[28]

We addressed this issue by also using RBCC (rule-based CC), a variant of KBCC that allows symbolic rules to be injected into a source network as in the knowledge-based artificial neural networks (KBANN) algorithm.[29] Rules are injected, not by learning via gradual weight adjustments, but rather by fixing the weights to appropriate values. An advantage of rule injection in RBCC is that rules can be represented as crisply as desired. Alternatively, if rules are inductively learned by a CC source network, they will not likely be as crisply represented. The ability to receive and use symbolic rules could be important for robots learning by direct instruction from humans or other robots.

RBCC automatically constructs a neural network implementation of a given rule through an n-of-m scheme in which n of m inputs can turn on the output unit.[30] This is more general and more flexible than using only AND (m-of-m) and OR (1-of-m). Translating a single rule into a neural network was performed using a two-layer network with as many input units as rule conditions, and a single output unit representing the action (or conclusion) of the rule. In this network, each weight was set to $W = 4$ or -4 (depending on whether the antecedent is negated or not).

Using such high weights tended to saturate the output unit, making for rather crisp rules. The bias weight Ω, from an input unit that is always on with an activation of 1, was set to:

$$\Omega = (m + 1 - 2n)W. \tag{4}$$

Unlike KBANN networks that involve hard coding all possibly relevant rules into a target network,[29] RBCC itself decides which rule-based sources to recruit, based on the standard CC criterion of selecting the source whose output activations covary best with target-network error. We compared CC, KBCC, and RBCC to KBANN and to RBCC1,2 a version that encoded only rules 1 and 2 as sources, the two rules that KBCC had found to be most relevant. Twenty networks for each algorithm were trained on the splice junction problem using a ten-fold cross-validation design.

Mean results on generalization and learning speed are plotted in Fig. 6. One-way ANOVAs revealed significant main effects of algorithm on each measure. The thick vertical bars segregate homogeneous subsets of means that do not differ significantly from each other. On the generalization measure, CC and KBCC did significantly better than KBANN, and significantly worse than RBCC and RBCC1,2. On learning speed, KBANN was significantly slower than KBCC and RBCC, which in turn were significantly slower than CC, which was slower than RBCC1,2.

On this problem, then, knowledge improved learning only if the algorithm decided which knowledge to recruit, a crisp representation of rules helped learning more than learning the rules from examples did, and learning was faster when knowledge contained only the most relevant rules. Compared to previous methods such as KBANN, KBCC and RBCC are more autonomous in being able to recruit the rules they deem helpful, a clear advantage for robots that are supposed to function autonomously.

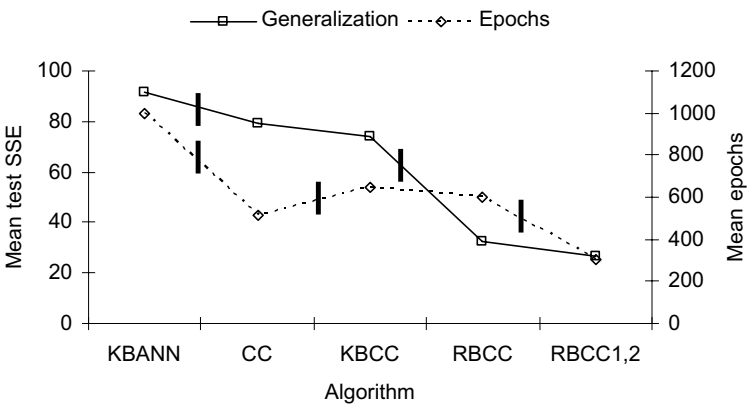


Fig. 6. Generalization and learning speed for five algorithms learning to identify DNA splice junctions.

3.3. *Psychology simulations*

A major goal of our laboratory is to apply KBCC to the simulation of psychological processes, and we are beginning to do that. Here we review two of these simulations, one on prime-number detection and the other on stages of the balance-scale problem.

3.3.1. *Primality*

The problem here is that of prime-number identification. An integer greater than 1 is *prime* if it has exactly two divisors, 1 and itself. An integer greater than 1 having more than two divisors is termed *composite*. The integer 1 is by definition neither prime nor composite.

One might think that the primality of an integer n could be determined by checking whether n is divisible by any integers between 2 and $n - 1$. Indeed it can be, but such testing can be much more efficient. The only divisors needed are primes from 2 to the integer part of $\sqrt{n}$.[31] Further increases in efficiency can be gained by starting with the smallest prime and increasing divisor size until finding a divisor that works. Starting with small divisors and increasing divisor size is efficient because the smaller the prime divisor, the more composites it can detect in a fixed range of integers.

KBCC target networks learning to classify integers as prime or composite came to perform in just this fashion when their pool of source knowledge contained networks that knew whether an input integer could be divided by each of a range of divisors.[32] The candidate pool contained a divide-by-2 network, a divide-by-3 network, etc. up to a divisor of 20. These source networks were trained on integers from 2–360 with default parameter settings except for a low score threshold of 0.01 to ensure a high level of accuracy for knowledge of divisibility. Twenty KBCC target networks trained on 306 randomly-selected integers from 21-360, with default parameter settings, recruited only source networks involving prime divisors below the square root of the largest number they were trained on (360). They avoided recruiting single hidden units, source networks with composite divisors, any divisors greater than square root of 360 even if prime, and divisor networks with randomized connection weights. Moreover, they recruited their divide-by sources in order from small to large, installing all recruits on a single layer.

Installing more than one recruit on the same layer is enabled by a variation of CC called sibling-descendant cascade-correlation (SDCC).[33] One-half of the candidate recruits (known as siblings) have no inputs from the previous layer of hidden units; the other half of the candidates (known as descendants) do have such inputs just as in classical CC. Sibling and descendant candidates compete with each other to be recruited based on their relative values of G as computed in Eq. (2). In this fashion, SDCC can build a variety of network topologies, depending on which recruit's activations covary best with network error at the time of recruitment. Extension of the sibling-descendant idea to KBCC allows previously-learned networks to likewise be

installed as either siblings or descendants. The SDCC option was used here in both source and target network training.

To our initial surprise, KBCC target networks never recruited a divide-by-2 source network, but it turned out this was because they instead learned to use the least significant digit of n to directly determine if n was odd or even. As with humans who are likely to use the least significant digit to check for divisibility by 5 or 10, this is an effective shortcut to actually dividing by 2.

Developing in this fashion, KBCC target networks learned to classify their training integers in about one third of the epochs required by 20 knowledge-free control networks, with fewer recruits on fewer network layers, and they generalized almost perfectly to the 34 novel test integers. In contrast, even after learning the training patterns to perfection, knowledge-free networks generalized less well than automatic guessing that the integer was composite, which was true 81% of the time for the integers employed. Mean epochs to learn and percent accurate generalization are double-y plotted in Fig. 7. ANOVAs revealed that all mean differences were significant, $p < 0.001$.

A knowledge-representation analysis of the KBCC networks further revealed that they had composed an understanding of primality characterized by the Boolean expression: $\neg(n \text{ is divisible by } 2) \wedge \neg(n \text{ is divisible by } 3) \wedge \cdots \wedge \neg(n \text{ is divisible by the largest prime} \leq n)$. In other words, these KBCC networks represented primality as a composition of divisor components, thus contradicting a popular view that compositionality is beyond the ability of artificial neural networks.[24,25] Because the internals of the recruited source networks were preserved, it was argued that this type of compositionality is fully *concatenative*, unlike the mere *functional* compositionality of artificial neural networks that are unable to recruit existing knowledge.[34,35] As predicted from the KBCC simulation,[32] new psychological research shows that

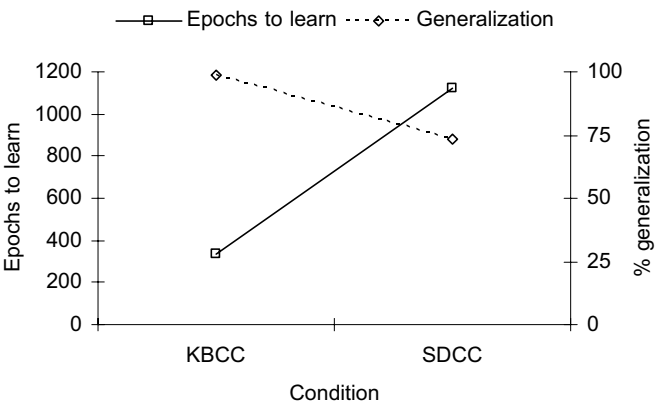


Fig. 7. Learning speed and generalization in knowledge-based (KBCC) and knowledge-free (SDCC) learning of primality detection.

university students testing the primality of integers also used mainly prime divisors below $\sqrt{n}$ and ordered their divisors from small to large.

Detecting prime numbers is not a task that many robots will likely have to face. However, the task is of interest because it requires prior knowledge in order to learn. Even though knowledge-free SDCC networks mastered all of the training patterns, they could not generalize better than guessing *composite*, the more common option. In contrast, KBCC networks generalized almost perfectly as noted. Because there are likely to be other robotic tasks that require knowledge in order to learn categories, KBCC or something like it may be required. Once again, the learning speedup for KBCC was considerable.

3.3.2. *Cognitive development on the balance-scale problem*

Here we illustrate knowledge-based learning in the simulation of cognitive development on the balance-scale scale task, one of the most widely simulated problems in developmental psychology. For this task, a child or adult is presented with a rigid beam balanced on a fulcrum.[36] The apparatus is designed with several pegs positioned on the beam at regular intervals to the left and right of the fulcrum. An experimenter places a number of identical weights on a peg on the left side and some number of weights on a peg on the right side of the scale. The participant is asked which side of the scale will descend, or whether the scale will remain balanced, when the beam is released from its moorings, usually a block placed under each end of the beam.

A rule that yields a correct answer to every such balance-scale problem involves multiplying the weight and distance from the fulcrum on each side and picking the side with the larger product (or torque) to descend. Only a few balance-scale problems actually require such torque computations for a correct solution, one of which is shown in Fig. 8(b). Most balance-scale problems can be solved by developmentally more primitive rules such as picking the side with the larger sum of weight plus distance values [see Fig. 8(a)]. Many balance-scale problems can be solved by even more primitive rules, such as picking the side with more weights or the side with the weights placed farther from the fulcrum.

Older simulations showed that CC networks could capture most of the stages seen in children, namely picking the side with more weights, then also picking the side with greater distance when the weights were equal on both sides, then being

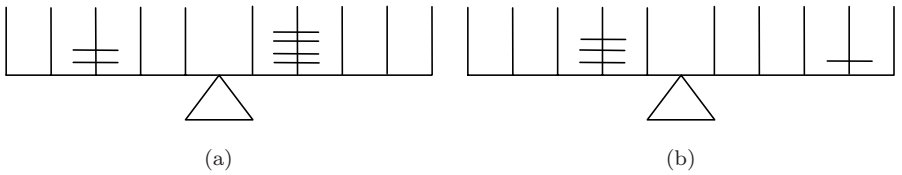


Fig. 8. Sample balance-scale problems solvable by (a) an addition rule or (b) a torque rule.

correct when weight and distance cues both predicted the same result and performing at chance when these cues conflict as in the problems shown in Fig. 8.[37] New simulations show that there are two ways for constructive neural networks to acquire the most advanced, torque rule. One method is to learn from balance-scale examples without the benefit of prior knowledge, as with CC or SDCC. The other method is to recruit an explicitly-taught torque rule with KBCC, which allows quick mastery of all balance-scale problems.

Our experience teaching university students about psychological development on the balance scale is that those few students who spontaneously use the torque rule to solve these problems admit that they learned this method in science classes, either in secondary school or college. Once the remaining students have been informed that balance-scale problems can be solved by computing and comparing torques, they too routinely use this rule to produce correct answers. Because this quick lesson requires neither multiplication training nor numerous examples, it likely relies on explicit instruction that builds on students' already existing knowledge. Because problems requiring the torque rule [like that in Fig. 8(b)] are so rare, learning from examples may not be a realistic way to learn about torque. It seems more likely that most people learn a torque rule, not from processing many examples, but rather from explicit verbal instruction. Knowledge-free simulations that learn from scratch with CC or SDCC confirm that it is difficult to capture all four balance-scale stages seen in children, particularly if the terminal stage 4 requires correct performance on torque problems such as in Fig. 8(b).

Knowledge-based learning with KBCC performs better, particularly in making the transition from the intuitive responding characteristic of the first three stages to the uniformly correct responding characteristic of the torque rule in stage 4. 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 steep sigmoid output unit. Such a function alternatively could be implemented in purely network form by using, say, product units that multiply rather than add their inputs.[38] KBCC can equally well recruit functions or networks, the only restriction being that they can be represented as differentiable functions.

In Fig. 9, we plot the mean proportions of training problems (a randomly-selected mixture of addition and torque problems) and test problems (other randomly-selected addition and torque problems) that 20 KBCC networks with default parameter settings got correct when trained for different numbers of epochs. It is noteworthy that adequate performance on addition problems is achieved before adequate performance on torque problems. It was not until the torque-difference function was recruited, at around 75–100 epochs that networks did well on torque problems.

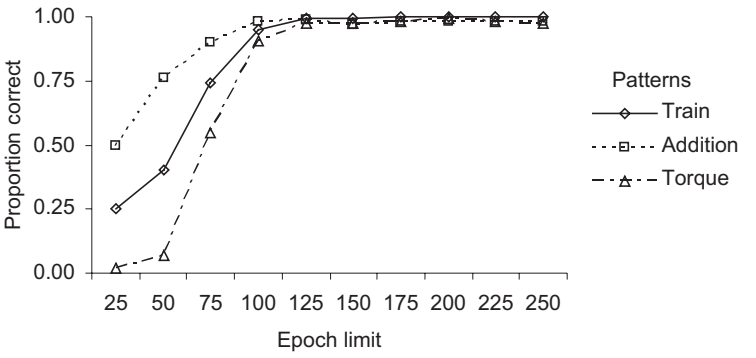


Fig. 9. Proportion correct in 20 KBCC networks trained for different amounts on a mixture of addition and torque problems and tested on each of these types of problems.

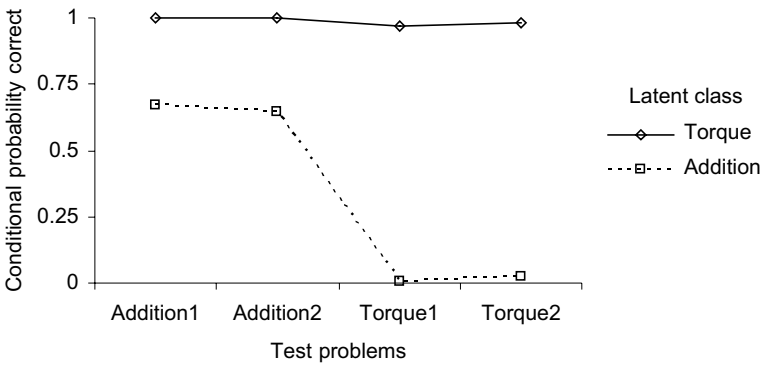


Fig. 10. Conditional probability of KBCC networks being correct on addition and torque test problems given membership in stage 3 (addition rule at about 50 epochs) or stage 4 (torque rule at about 125 epochs) on the balance-scale task.

One way to diagnose rule use, whether in networks or in children, is to subject the distribution of response patterns, weighted by their frequency of occurrence, to Latent Class Analysis (LCA).[39–41] In what is called exploratory LCA, the estimated parameters of a statistically-fitting model differ across latent classes, typically designating homogeneous groups of people (or networks) that differ from other groups.[42] Estimated parameters can then be compared across classes to determine how the classes differ from each other. Finally, the cases can be sorted into the latent classes based on membership probabilities estimated from the model. Results from 250 KBCC networks trained on randomly selected addition and torque problems for up to 50 epochs and 250 KBCC networks similarly trained for up to 125 epochs, all with default parameter settings, are presented in Fig. 10 in terms of the conditional probabilities of being correct on addition and torque test problems given membership in the addition-rule or torque-rule class. Networks trained for up to 50

epochs show a pattern of performance characteristic of using an addition rule — they are likely to be correct on addition problems but not on torque problems. In contrast, networks trained for up to 125 epochs are very likely to be correct on both torque and addition problems. This pattern roughly simulates children's development from what some balance-scale researchers regard as a kind of advanced stage 3 performance to stage 4 performance.

Such results, coupled with the difficulty of capturing all stage transitions without direct instruction in torque, support our intuition that knowledge-based learning may be required to cover balance-scale development. Knowledge-based learning with KBCC shows how explicit instruction on how to compute and compare torques can quickly lift a computational system to consistently correct performance without implementing any major change in training patterns. This is in accord with the rather sudden and late effects of explicit instruction on torque, as sometimes happens in school science classes. Although further investigation is warranted, acquisition of the final stage of balance-scale performance appears to be a case where knowledge-based learning is more psychologically plausible than the extensive knowledge-free learning from many examples that simulates transitions through the first three stages.

In analogous fashion, innate knowledge, which some argue to be essential for subsequent learning and development,[43–45] conceivably could be simulated by injection into KBCC source networks for possible recruitment by KBCC target networks.

The balance-scale problem *per se* is not likely to be important to most robots, but it has been the primary benchmark problem for simulations of cognitive development for some years. Moreover, the balance-scale problem is representative of a large number of problems that involve integrating multiple sources of quantitative information, here weight and distance, to make a qualitative decision, in this case on the behavior of the beam. Many robots are likely to face such information integration problems. They also may sometimes require direct instruction on use of functions or perhaps the ability to recruit innate knowledge of functions.

4. Design of Candidates

The source networks that serve as candidate recruits in KBCC have so far been designed to serve certain goals that governed the research. In most cases, source networks were not directly designed, but were instead learned by CC or SDCC networks. In principle, they could also be learned by KBCC networks in a recurrent manner. The key ideas governing the design of the tasks on which source learning was based were knowledge relevance, component simplicity, and complexity control.

In our early work on geometric shapes, we wanted to determine the effects of certain source knowledge on learning, as measured by learning speed and generalization. This required source tasks that were homogeneous within conditions and which

varied across conditions. We also wanted to know how KBCC chose among multiple candidate sources. This goal required heterogeneous sources within conditions. In all cases, KBCC, like CC and SDCC, recruits the candidate whose output activation best covaries with current network error. Studies with homogeneous candidate pools revealed that networks learned faster with relevant than with irrelevant sources. In many cases, relevance could be defined *a priori* as knowing a component part of the learning target.

Experiments on the balance-scale task were interested in whether direct explicit instruction on the torque rule could be injected into a source network which was then recruited as needed. Such symbolic functions can indeed be injected into a source network and recruited by a target network as long as the function is differentiable. If sufficiently relevant to the target task, such source functions are in fact recruited.

Later the issue of controls for complexity came to the fore in KBCC research. We had noticed a general tendency for KBCC to recruit a network over single-unit candidates, and in some cases, complex over simple network candidates. Was this a preference for relevant knowledge or mere computational complexity? To answer this question, later studies included control sources that were as complex as those representing relevant knowledge, but which had randomized connection weights to remove relevant knowledge. Generally, mere complexity was not sufficient for high covariation with network error or for speeding learning or facilitating generalization.

If KBCC is used for robotics, it is likely that task sequences will be determined more by the natural features of the environment than by issues of relevance and control. In these cases, it will be interesting to see what gets recruited and in what order. Given a particular set of candidates, order of recruits is always determined by the KBCC covariation criterion, and not by the experimenter. This provides yet another sort of autonomy for systems using KBCC.

Ordering of source tasks has not been much studied, but could be of both practical and theoretical interest. There might be interesting implications for robot curriculum design because some orders may facilitate learning more than others do. We would expect that the principles of relevance and component simplicity would be operative here. A complex lesson could be broken down into component parts and these parts could be learned by source networks and then recruited by a target network. In the case of complex tasks, such as primality testing, lesson structuring not only facilitated learning but was also necessary for learning. Researchers who are training robots in other difficult tasks may want to structure subtask orders carefully rather than allowing learning to proceed in a more spontaneous or chaotic fashion.

5. Examples of Difficult Learning Problems Meriting a Knowledge-Based Approach

We next suggest the likely importance of knowledge-based learning in two additional domains that have been well studied in the psychological literature and are

of obvious interest to developmental robotics — visual object recognition and word learning.

5.1. *Visual object recognition*

The problem here is one of activating a memory representation of a stimulus category from a visual image projected on the retina. Biederman[46] devised an ingeniously simple technique for estimating how many objects humans can name on the basis of their visual shape alone. He first counted the mean number of dictionary entries referring to familiar objects that can be identified by shape on each of a random sample of pages, excluding those items that are identified by surface properties such as color or texture, and then multiplied that value by the total number of dictionary pages. This yielded 1,600 terms, which he doubled to allow for idiosyncratic classes and objects not captured by the dictionary sampling, yielding a rough estimate of 3,000 object classes. Then, reasoning that each of these classes has about ten perceptual models, he estimated 30,000 object models for the typical competent adult.

Because the typical 6-year-old has achieved adult competence in naming visual objects, and has been awake for about 30,000 hours, Biederman estimated that a typical child learns new object models at the average rate of one per waking hour. This would be an exceedingly difficult learning problem if every new object class had to be learned from scratch, as if nothing relevant was already known. The problem of learning object classes would be much more tractable for a system that built its new learning of object classes on top of already known object properties.

Biederman's[46] recognition-by-components theory does just that, by representing an object as an arrangement of a small number of simple primitive volumes (called geons) and the relations among them. Geons are members of a particular set of convex or singly concave volumes that can be modeled as generalized cones. They have the computational advantage that they can be distinguished from almost any viewpoint and are resistant to visual noise. Biederman estimates that a geon-equipped system can recognize about 10.5 million objects composed of two geons and over 306 billion three-geon objects. The basic idea is that, if an arrangement of two or three geons can be recovered from an image, then objects can be quickly recognized even if they are novel, occluded, rotated, degraded, or lack detail, color, and texture.

Hummel and Biederman[47] presented a hard-coded connectionist simulation of this recognition-by-components theory of object recognition in which there were seven distinct layers of units: edges; vertices, axes, and blobs; geon attributes; two layers for geon relations; geon feature assemblies; and finally objects.

More abstractly, one might imagine a general knowledge-based learner such as KBCC that automatically abstracts the basic components of visual objects from the first few learning opportunities and then uses these components to learn new object classes. Such a scheme was implemented within a Bayesian framework in which

object shape was represented by a joint Gaussian distribution of the locations of object features.[2]

5.2. *Word learning*

Another example of the difficulty of learning across childhood is that of learning words, the arbitrary pairing of a word root with a sequence of sounds.[43] Up until about 1.5 years, word learning is quite slow. But number of words doubles over the next three months and doubles again over the following three months. By six years of age, a typical child understands 10,000 words and by ten years 40,000 words.[48] Over this age period, children add more than 10 words per day to their vocabulary.[49] The typical high-school graduate is estimated to know about 45,000–60,000 words.[50]

This rapid and accelerating pace of word learning implies that not many exposures to a new word are required, and that new learning must be taking advantage of existing knowledge. Regarding number of exposures, evidence indicates that 1-year-olds require no more than ten exposures to a new word,[51] and that 2-3-year-olds require only a single exposure.[52] This is despite the fact that a helpful teacher saying, for example, "rabbit" when a rabbit hops past could mean any number of things: fur, floppy ears, a member of the species, fast moving, etc.[53] Among the constraints proposed to explain how a child hones in on accurate word meanings so quickly are that a new word refers to a whole object, to other objects of the same class, and to a novel object.[54] Such constraints can be understood as knowledge about what new words are likely to refer to.

The morphological regularities of a language allows creation of new words out of old words.[43] For example, the suffix *-able* converts a verb meaning to *perform an action* into an adjective meaning *capable of having this action done to it*, as in *learn/learnable*. There are many such derivational suffixes in English: *-ate*, *-ize*, and *-ly*, etc. The suffix *-er* converts a verb into a noun, as in *learn/learner*. And the suffix *-ness* converts an adjective into a noun, as in *heavy/heaviness*. Similarly, prefixes such as *pre-* and *un-* signal meaning changes involving *before* and *nullifying* as in *date/predate* and *happy/unhappy*.

Even word stems can be built of parts, as in compound words, like *toothbrush* and *heartbroken*. The ability of neural networks to learn the subtleties of how to convert plural nouns into compound nouns[55] has been demonstrated.[56] Understanding new words can also depend on knowledge of old words as in dictionary-style definitions that are often used by parents or others to explain what a new word means. The general point is that word learning builds on previous knowledge of words and meanings. These arguments suggest that any robot that would learn to converse with humans or with other robots should be able to learn quickly and to build current learning on existing knowledge. Once again, this is the sort of thing that KBCC could do. There are many additional examples in the psychological literature that illustrate the benefits of knowledge-based learning.

6. Relation to Previous Work on Knowledge-Based Neural Learning

As previewed, work on KBCC, RBCC, and FBCC is related to previous neural-network research on knowledge transfer, multitask learning, lifelong learning, sequential learning, knowledge insertion, modularity, and input re-coding.

Pratt[16] pioneered the technique of discriminability-based transfer to re-use weights from a previously trained network to initialize a new network. This may be the most obvious and straightforward way to use existing knowledge to aid new learning. She further improved this technique by re-scaling the previous network's hyper-planes so that the more useful ones had larger weights than the less useful ones.

Caruana[17] developed Multitask Learning (MTL) in which he trained in parallel a network on several tasks taken from the same domain, using a single output for each task. Such networks tended to learn a common hidden-unit representation, which often proved useful for learning subsequent tasks within the same domain. Baxter[57] showed that the number of examples required for learning any single task with MTL decreases with an increase in the total number of tasks learned in parallel.

Thrun and Mitchell[18] developed a technique called lifelong learning, in which a network learns the slope of a target function at each training example. Then, in new learning, a meta-network provides slope predictions and estimates its accuracy for each new training example.

Silver and Mercer[19] proposed the Task Rehearsal Method (TRM) as a way of enhancing sequential learning. In their scheme, old tasks are pseudo-rehearsed during new learning, essentially by generating patterns that are added to those of the target task. In pseudo-rehearsal, a network creates its own target vectors, using its current weights, rather than passively accepting target vectors from the environment.[8]

Shavlik[20] devised the KBANN algorithm for creating knowledge-based artificial neural networks. A set of symbolic rules embodying knowledge of a problem is converted into a programmer into a feed-forward neural network with the final rule conclusions as output units and intermediate rule conclusions as hidden units. Connection and bias weights are initialized to implement the logical structures of the rules. Networks so initialized with this domain knowledge are then trained with examples in order to further refine the knowledge. Training is typically faster and generalization is better than with standard networks that are initialized with random weights. After the training, symbolic renditions of the current rules can be extracted from the network.

Jordon and Jacobs[21] proposed the Hierarchical Mixture of Experts (HME) algorithm, designed to decompose a problem into separate network modules. Each of these network modules might become an expert on some subtask, and then cooperate on an overall solution via gating networks that learn to weight the contributions

of the modular experts. HME learned the dynamics of a four-degree-of-freedom robot arm faster than a multilayer backpropagation network did.

Clark and Thornton[22] discussed the importance of networks being able to recode their inputs in the learning of difficult problems that they called Type-2 problems. Type-1 problems are those that can be solved by processing the originally coded inputs. But Type-2 problems need to be recoded in order to become learnable by systems that have learned Type-1 knowledge. The ability to do this recoding would presumably require incremental learning, modularity, and perhaps representational redescription,[58] but no particular algorithm was proposed to accomplish this. Pfeifer and Scheier[59] envisioned two ways of translating Type-2 problems into Type-1 problems: using current knowledge as suggested by Clark and Thornton and evolved sensory motor interaction as illustrated by studies of simulated robots.[60,61]

In contrast to these previous methods for using knowledge in learning, KBCC uses established techniques from constructive neural learning algorithms.[15] KBCC recruits existing networks in addition to single units as it needs them in order to increase its computational power in the service of error reduction. Treating its existing networks just like untrained single units, KBCC trains weights to the inputs of its source networks in order to determine whether their outputs covary with error in the target network. KBCC also trains the output weights from its recruits in order to incorporate them into a solution of the target problem. This adaptation of the inputs and outputs of recruits gives KBCC considerable flexibility in its use of knowledge, allowing KBCC to use knowledge that is only partly relevant to the new target task. The fact that units and sub-networks can be installed in a cascade enables KBCC to build its new learning on top of any recruited knowledge. Alternatively, KBCC can use the SDCC option to build parallel solutions, or some mixture of parallel and cascaded solutions.

A direct comparison on translation of two-dimensional shape problems showed that KBCC learned considerably faster than MTL, which did not actually learn any faster with knowledge than without.[62] As noted, MTL failed to speed the learning of new tasks because it requires that both old and new tasks be freshly learned in parallel. In contrast, KBCC can use its existing knowledge without having to relearn it.

Unlike KBANN,[20] where all relevant rules are injected into a target network by an experimenter, KBCC more autonomously determines which rules (injected into source networks by an experimenter) are to be recruited into the target network. These injected rules compete with each other and with single hidden units and any learned source networks for being recruited. The candidate whose activations covary best with network error is the one that gets recruited at any moment. This selective recruiting often creates a smaller and more effective solution than does forced injection of all possibly relevant rules.[30]

KBCC implements a natural resistance to retroactive interference that often plagues sequential learning in neural networks.[63] Because each source network is a frozen module, it never loses its original functionality, no matter how many times and in how many ways it gets recruited. As noted, there is also no need for KBCC to relearn old tasks while learning new ones, as in both TRM[19] and MTL.[17]

Importantly, unlike many of the other knowledge-based neural learners, for which both inputs and outputs of the source and target task must precisely match, KBCC can potentially recruit any sort of function to use in a new target task. Source network inputs and outputs can exist in different numbers, be arranged in different orders, and use different coding techniques than the inputs and outputs of the target network. The only constraint on what can be recruited by KBCC is that the recruit must be a differentiable function. This requirement is due to the use of the Quickprop algorithm for weight adjustment.[23] If numerical estimation or an optimization algorithm that did not require differentiation were used, even this constraint would disappear. This flexibility ensures that functions created by means other than KBCC learning could be recruited, as in the variant RBCC[30] and FBCC[64] methods. This wide range of candidates would appear to offer considerably more power and flexibility than knowledge-based neural learning algorithms typically provide.

When a source network is recruited within KBCC, it is thereafter treated as a kind of black box module. Somewhat like the modules discussed by Fodor,[65] KBCC's recruited networks are computationally encapsulated subsystems that interact with the rest of the computational system only through their inputs and outputs. Although Fodor had proposed that such modules are innate and operate only in particular domains like perception and language, it has been more recently acknowledged that computational modules can be learned and can operate within central cognition[58] as KBCC does.

The computational advantages of network modularity have been well recognized.[66] In contrast to large, homogeneous networks, modular neural systems like KBCC split a task into parts, restrict complexity to be proportional to problem size, incorporate prior knowledge, generalize effectively, learn multiple tasks, perform robustly, and are relatively easy to modify. Also the solutions learned by modular networks are often easier to analyze than those learned by homogeneous networks. In KBCC, for example, whatever recruited modules do with their input does not change from the time of their initial acquisition, although it might remain challenging to determine their role in the overall solution reached by the target network. Unlike the HME approach to modularity,[21] KBCC's recruited networks are gradually constructed through automatic learning rather than being designed by an experimenter and being simultaneously present throughout training, thus providing additional autonomy.

KBCC is able to reduce a complex problem to a simpler, already known problem, as recommended by Clark and Thornton.[22] When a source network is recruited within KBCC, this effectively reinterprets a target problem as if it were an instance of an already known problem, or at least partially related to a known

problem. Additional recruitments and output-weight adjustments then craft this reinterpretation into a solution to the target problem. This could supplement the evolved-sensory-motor-interaction method already in use with robots,[60,61] but without the prolonged evolution required by genetic algorithms. Or perhaps KBCC could provide a cheaper way of acquiring the sensory-motor strategies of approaching, circling, and examining objects that aids category learning in robots. Our prime-number detection experiment shows that a Type-2 problem, too difficult for one of the most powerful knowledge-free algorithms (SDCC) to learn, can be reinterpreted by KBCC in terms of existing Type-1 knowledge.

In short, KBCC allows for a combination of learning by analogy or induction, and various combinations of analogy and induction. KBCC learns by analogy to its current knowledge whenever it can and switches to induction if it needs to. Recruiting a network is a case of learning by analogy, whereas recruiting single units and adjusting connection weights constitute learning by induction. Both processes are seamlessly integrated within KBCC learning.

7. Relation to Previous Work on Analogical and Case-Based Reasoning

KBCC learning also bears some relation to both analogical[67] and case-based.[68] reasoning. Reasoning by analogy or past cases involves finding a relevant source analogy (or case) in memory, mapping this source to the current target problem, tweaking the mapped source to better fit the target problem, using the tweaked source to solve the target problem, and finally storing the new solution in memory.

KBCC differs from these approaches in two principal ways: KBCC focuses on learning rather than reasoning and on neural rather than symbolic representations. KBCC finds a source with the highest correlation with current network error, it maps and tweaks the current problem onto this and possibly other sources by adjusting connection weights from the target inputs to the source inputs, determines how to use the recruited source by adjusting output weights from the source, and stores the new solution as a whole network including connection weights. It may be interesting to explore relations between KBCC and reasoning by analogy or cases. KBCC implements some of the steps involved in these reasoning methods in novel, probably less brittle, ways, and it might benefit from more ability to perform reasoning based on these learned representations.

8. Neurological Correspondence

Although not a detailed model of brain circuits, KBCC is inspired by evidence about how the brain supposedly works and thus incorporates many neurological features. Like other artificial neural networks, KBCC contains generic neurons with sigmoid activation functions having an activation floor and ceiling and a sharp, but continuous threshold between them. Like ordinary CC, KBCC implements both

direct and indirect connectivity between sites and grows by recruiting new computational devices, suggestive of synapto- and neuro-genesis.[69] The topology of a KBCC network changes, not only by growing, but optionally also by pruning relatively unused connections.[70] KBCC also mimics two fundamental features of brain organization, functional specialization and integration.[71] Source networks in KBCC are typically specialized and can be integrated into solutions of new tasks through learning. Unlike a variety of hybrid systems for combining symbolic and neural methods,[72–75] RBCC is implemented in a homogeneous neural fashion, as are brain networks. RBCC has been used to model the interactions between frontal and temporal cortices during resolution of lexical ambiguities.[76]

In both humans and non-human primates, the selection of prior knowledge representations involves the left prefrontal cortex (LPFC).[77–79] In a similar way to the workings of KBCC, multiple competing representations can be activated in parallel within the LPFC. Once a candidate rule or concept is selected, the LPFC plays a central role in its maintenance within working memory. In this way, the activation of a selected representation can be maintained as long as required for performing a given task. Analogously, KBCC sustains the flow of activity that a selected source network sends and receives for the duration of the task, thus ensuring the participation of relevant prior knowledge in the solution learned by the target network.

9. Discussion

In closing, we address some of the advantages and current limitations of knowledge-based learning as implemented in KBCC, and indicate some avenues for future research.

9.1. *Advantages of KBCC*

KBCC is a general learner with considerable power and flexibility. Like CC, KBCC can automatically construct a network to suit the particular problem being learned, and escape from Fodor's paradox about not being able to learn anything genuinely novel.[80,81] It escapes from Fodor's[82] paradox by building new computational power that allows for representation and learning of hypotheses that it could not previously represent. KBCC seamlessly integrates inductive and analogical learning such that each complements the other. It learns by analogy to what it already knows whenever it can, resorts to learning by induction from examples when it knows nothing relevant, and achieves required combinations of learning by analogy and induction.[4]

Existing knowledge is automatically selected from various sources without the intervention of a human experimenter and without regard for the number and order of network inputs and outputs.[4] KBCC selects, maps, and tweaks existing knowledge automatically to aid new learning, and is able to compose solutions without

changing the recruited components,[32] as in classical, concatenative compositionality. Building on existing knowledge in these ways allows learning to be fast and accurate[4] and is sometimes necessary for neural learning to generalize.[32]

9.2. *Limitations of KBCC and possible future work*

One of the biggest limitations of KBCC is that its search for source knowledge is computationally expensive, especially in a mature and experienced network that has learned a lot. However, the difficulty that humans have in finding analogous knowledge that they are known to possess suggests that this limitation may actually enhance data coverage in psychology simulations. Humans are known to sometimes require hints in their search for analogous knowledge.[83,84] Perhaps KBCC could similarly benefit from hints, conceivably by biased weighting of Gs in Eq. (2) that are associated with various source networks.[76] Methods for effectively pruning the candidate search space are under active investigation.

Imitation of others is another potentially fruitful way to find relevant knowledge. Research indicates that performing basic motor actions and observing these actions performed by another agent activate the same neurons.[85] Such *mirror* neurons thus represent a mechanism by which existing procedural knowledge can be activated in learning a novel sequence of actions demonstrated by another actor. Imitation can occur at various levels, from program level imitation at the highest level of abstraction down to the action level where detailed behavior is specified.[86] Despite the fact that true imitation can be confounded with simpler cognitive mechanisms such as priming and facilitation, even those simpler mechanisms could help locate relevant knowledge during new learning. Whether imitation can be encompassed by KBCC is currently under study. If robots could learn by imitating humans or other robots, this could increase their autonomy and lessen the need for explicit instruction.

At this point, there are still too few psychology simulations using KBCC. This is partly due to the fact that most psychology experiments on knowledge-based learning use quite simple linearly-separable problems.[9,10,14] KBCC, by starting in output phase, can learn such problems without having to recruit any knowledge. A possible solution might involve starting KBCC in input phase instead, where it would try to recruit relevant knowledge even for simple linear problems. Another solution would be to continue working with complex non-linear problems and bring these into the laboratory for psychological study, as illustrated here with primality.

Another limitation is that, in its current form, KBCC never modifies the source networks that it recruits. Only the input weights to the source and the output weights from the source get modified with learning. Freezing of internal connection weights and installation of copies of source networks are computationally effective, but may be psychologically unrealistic. For one thing, it means that KBCC cannot

capture retroactive interference, in which new learning interferes with old knowledge, even though KBCC can capture proactive interference. Freezing of internal connection weights also seems to conflict with psychological research on memory consolidation indicating that memories are labile just after retrieval.[87] A natural way to modify weights inside of source networks would be to back-propagate error signals through all layers of source networks. It is an open question whether this would damage KBCC learning and performance and cover emerging results on memory reconsolidation.

Still another problem is that the technique currently used in pruning KBCC (and CC) networks is not psychologically realistic. In this technique, weights are pruned from a network until error on an additional, generalization test set begins to increase,[70] a method known as *early stopping*. This is unrealistic because such test sets rarely occur in natural human learning. Perhaps simpler pruning methods such as removing very small weights would work nearly as well and be more psychologically realistic.

Another current limitation of KBCC is that segmentation of learning tasks is done by a human experimenter. It would be more natural for a learning algorithm to autonomously determine whether a learning task is new or old. This is a problem that is common in machine-learning algorithms and it has received relatively little attention. Perhaps the passage of time, changes in content or context, or changes in input or output coding could signal that a learning task is new and thus requires a fresh target network. An automatic novelty detector might also help with this problem.[88]

Finally, KBCC, like other neural networks, can become trapped in a local minimum while attempting to reduce error. KBCC addresses this issue by recruiting new hidden units and previous knowledge during training which effectively changes the landscape of the error surface, often allowing a more productive path to lower error. However, there is no guarantee that recruits will be ideal. Recent techniques based on incremental learning (e.g. IHDR[89]) address the local minimum problem by building a hierarchical distribution tree. Although generally speaking tree-based systems have not faired as well as CC in capturing cognitive development, they do nonetheless reproduce some developmental effects.[7] It might be fruitful to combine KBCC's efficient knowledge transfer capabilities with the learning capabilities of IHDR. How this can be achieved remains to be explored.

9.3. *Some robotics issues*

There are also some issues relevant to robotics that would need to be addressed. KBCC is a deterministic learning algorithm that requires target vectors to supervise its learning. However, robots must cope with complex environments that are stochastic and also often lack a knowledgeable teacher. To deal with stochastic uncertainty, KBCC could be supplemented with Q-learning to evaluate the quality of particular actions in particular environmental states[90] or stochastic units that

output binary values with learnable probabilities.[91] Q-learning and other reinforcement learning techniques require only scalar reward signals, which may be more common in natural environments than fully specified target response vectors are. Some progress in integrating KBCC with temporal-difference reinforcement learning has already been reported.[92]

In many cases, natural environments may even supply target vectors. One such example that we discussed earlier is that of the balance scale, where a prediction of which way the scale will tip (implemented by neural outputs) can be compared to which way it actually tips (effectively a target vector) to generate an error signal that can be used to adjust connection weights. More generally, the discrepancy between predictions about what will happen next in the environment and what actually happens can be used to generate error signals and thus self-supervise learning. Thus, characterizing knowledge-based neural networks as non-starters in robotics applications because they are deterministic or supervised seems overly narrow.

Another problem is that robots must learn and perform in real time, but even simple learning algorithms can take much longer. As we amply demonstrated here, learning can be much faster when it is based on existing knowledge, as in KBCC. Whether knowledge-based learning will be fast enough for robotic applications remains to be seen. Already though, there is some evidence that neural-network learning can play a useful role in robotic applications. For example, the CES (C++ for Embedded Systems) robotic software system integrates neural-network learning with probability distribution data structures.[93]

There has been a tension in the robotics literature between approaches that emphasize deliberate planning versus those that emphasize emergent reactivity.[94] Some hybrid systems successfully combine reaction with deliberation. One of the most popular hybrids is a three-layer architecture combining reactive, executive, and deliberate processes.[95] KBCC would more likely operate at the upper two layers than at the reactive layer in such an algorithm.

KBCC and related algorithms are currently implemented in disembodied artificial neural networks that have no contact with the real world. These are general and powerful algorithms for learning, generalizing, and building on existing knowledge, but more research would be needed to study their utility in a robotic context. These general learners would need to interact with a robot's sensors, actuators, and control system so that the robot could learn, represent, and retrieve knowledge that these systems require to interact in real time with the physical and social world. Whether these integrations could be achieved constitutes a significant yet worthy challenge for future research.

10. Conclusion

Developmental robotics strives to create robots equipped with the human abilities that allow them to learn in complex environments. An important human capacity that robots often lack is the ability to use their knowledge to guide new learning.

Here we showed how KBCC, a knowledge-based neural learner, might be useful in such applications. KBCC autonomously recruits its previously-learned networks as well as single hidden units to create its own network topology, enabling learning by analogy when possible and learning by induction when required. Applied to a variety of toy, realistic, and psychology simulation problems, KBCC illustrates that knowledge recruitment both speeds and enables learning. The current limitations of such systems point to possibly fruitful pathways for new research.

Acknowledgments

This research was supported by a grant from the Natural Sciences and Engineering Research Council of Canada to Shultz, an FQRNT Graduate Fellowship to Thivierge, and a McGill Major Fellowship to Dandurand. We are grateful to Yoshio Takane for some excellent comments on a previous draft.

References

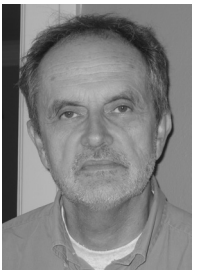
1. J. Weng, J. McClelland, A. Pentland, O. Sporns, I. Stockman, M. Sur and E. Thelen, Autonomous mental development by robots and animals, *Science* **291** (2000) 599–600.
2. F.-F. Li, Knowledge transfer in learning to recognize visual object classes, in *Proc. 5th Int. Conf. Development and Learning ICDL 2006*, Department of Psychological and Brain Sciences, Bloomington, Indiana, USA (2006).
3. J. Weng, Developmental robotics: Theory and experiments, *Int. J. Humanoid Robot.* **1** (2004) 199–236.
4. T. R. Shultz and F. Rivest, Knowledge-based cascade-correlation: Using knowledge to speed learning, *Connect. Sci.* **13** (2001) 1–30.
5. N. Chater, J. B. Tenenbaum and A. Yuille, Probabilistic models of cognition: Conceptual foundations, *Trends Cogn. Sci.* **1** (2006) 287–291.
6. J. L. Elman, E. A. Bates, M. H. Johnson, A. Karmiloff-Smith, D. Parisi and K. Plunkett, *Rethinking Innateness: A Connectionist Perspective on Development* (MIT Press, Cambridge, MA, 1996).
7. T. R. Shultz, *Computational Developmental Psychology* (MIT Press, Cambridge, MA, 2003).
8. A. Robins, Transfer in cognition, *Connect. Sci.* **8** (1996) 185–203.
9. G. Nakamura, Knowledge-based classification of ill-defined categories, *Mem. Cogn.* **13** (1985) 377–384.
10. E. Heit, Models of the effects of prior knowledge on category learning, *J. Exp. Psychol. Learn. Mem. Cogn.* **20** (1994) 1264–1282.
11. F. C. Keil, Conceptual development and category structure, in *Concepts and Conceptual Development: Ecological and Intellectual Factors in Categorization*, ed. U. Neisser, (Cambridge University Press, Cambridge, 1987).
12. G. L. Murphy, A rational theory of concepts, *Psychol. Learn. Motiv.* **29** (1993) 327–359.
13. M. J. Pazzani, Influence of prior knowledge on concept acquisition: Experimental and computational results, *J. Exp. Psychol. Learn. Mem. Cogn.* **17** (1991) 416–432.
14. E. J. Wisniewski, Prior knowledge and functionally relevant features in concept learning, *J. Exp. Psychol. Learn. Mem. Cogn.* **21** (1995) 449–468.

15. S. E. Fahlman and C. Lebiere, The cascade-correlation learning architecture, in *Advances in Neural Information Processing systems 2*, ed. D. S. Touretzky (Morgan Kaufmann, Los Altos, CA, 1990), pp. 524–532.
16. L. Y. Pratt, Discriminality-based transfer between neural networks, in *Advances in Neural Information Processing Systems 5* (Morgan Kaufmann, San Mateo, CA, 1993), pp. 204–211.
17. R. Caruana, Multitask learning, *Mach. Learn.* **28** (1997) 41–75.
18. S. Thrun and T. Mitchell, Integrating inductive neural network learning and explanation-based learning, in *Proc. 13th Int. Joint Conf. Artificial Intelligence*, ed. R. Bajcsy (Morgan Kaufmann, San Mateo, CA, USA, 1993).
19. D. Silver and R. Mercer, The parallel transfer of task knowledge using dynamic learning rates based on a measure of relatedness, *Connect. Sci.* **8** (1996) 277–294.
20. J. W. Shavlik, A framework for combining symbolic and neural learning, *Mach. Learn.* **14** (1994) 321–331.
21. M. I. Jordan and R. A. Jacobs, Hierarchical mixtures of experts and the EM algorithm, *Neural Comput.* **6** (1994) 181–214.
22. A. Clark and C. Thornton, Trading spaces: Computation, representation and the limits of uninformed learning, *Behav. Brain Sci.* **20** (1997) 57–97.
23. S. E. Fahlman, Faster-learning variations on back-propagation: An empirical study, in *Proc. 1988 Connectionist Models Summer School* eds. D. S. Touretzky, G. E. Hinton and T. J. Sejnowski (Morgan Kaufmann, Los Altos, CA, 1988), pp. 38–51.
24. J. A. Fodor and Z. W. Pylyshyn, Connectionism and cognitive architecture: A critical analysis, *Cognition* **28** (1988) 3–71.
25. S. Pinker, *How the Mind Works* (Norton New York, 1997).
26. F. Rivest and T. R. Shultz, Compositionality in a knowledge-based constructive learner, in *Papers from the 2004 AAAI Fall Symp.*, Menlo Park, CA, USA (AAAI Press, 2004), pp. 54–58.
27. F. Rivest and T. R. Shultz, Application of knowledge-based cascade-correlation to vowel recognition, *IEEE Int. Conf. Neural Networks*, Hawaii (IEEE Press, 2002) pp. 53–58.
28. J. P. Thivierge and T. R. Shultz, Finding relevant knowledge: KBCC applied to splice-junction determination, *IEEE Int. Conf. Neural Networks*, Hawaii (IEEE Press, 2002) 1401–1405.
29. G. G. Towell and J. W. Shavlik, Extracting refined rules from knowledge-based neural networks, *Mach. Learn.* **13** (1993) 71–101.
30. J. P. Thivierge, F. Dandurand and T. R. Shultz, Transferring domain rules in a constructive network: Introducing RBCC, in *Proc. IEEE Int. Joint Conf. Neural Networks*, Budapest, Hungary (2004), pp. 1403–1409.
31. T. Nagell, *Introduction to Number Theory* (Wiley, New York, 1951).
32. L. Egri and T. R. Shultz, A compositional neural-network solution to prime-number testing, in *Proc. 28th Ann. Conf. Cognitive Science Society*, Vancouver, Canada (Erlbaum, Mahwah, NJ, 2006), pp. 1263–1268.
33. S. Baluja and S. E. Fahlman, Reducing network depth in the cascade-correlation learning architecture, School of Computer Science, Carnegie Mellon University, Technical Report, CMU-CS-94-209 (1994).
34. J. Pollack, Recursive distributed representations, *Artif. Intell.* **46** (1990) 77–105.
35. T. van Gelder, Compositionality: A connectionist variation on a classical theme, *Cogn. Sci.* **14** (1990) 355–364.
36. R. S. Siegler, Three aspects of cognitive development, *Cogn. Psychol.* **8** (1976) 481–520.

37. T. R. Shultz, D. Mareschal and W. C. Schmidt, Modeling cognitive development on balance scale phenomena, *Mach. Learn.* **16** (1994) 57–86.
38. M. Schmitt, On the complexity of computing and learning with multiplicative neural networks, *Neural Comput.* **14** (2001) 241–301.
39. J. Boom, H. Hoijtink and S. Kunnen, Rules in the balance: Classes, strategies, or rules for the balance scale task, *Cog. Dev.* **16** (2001) 717–735.
40. B. R. J. Jansen and H. L. J. van der Maas, Statistical test of the rule assessment methodology by latent class analysis, *Deve. Rev.* **17** (1997) 321–357.
41. B. R. J. Jansen and H. L. J. van der Maas, The development of children's rule use on the balance scale task, *J. Exp. Child Psychol.* **81** (2002) 383–416.
42. A. L. McCutcheon, *Latent Class Analysis* (Sage, Newbury Park, 1987).
43. S. Pinker, *The Language Instinct* (Harper Collins, New York, 1994).
44. E. S. Spelke, Initial knowledge: Six suggestions, *Cognition* **50** (1994) 431–445.
45. K. Wynn, Addition and subtraction by human infants, *Nature* **358** (1992) 749–750.
46. I. Biederman, Recognition-by-components: A theory of human image interpretation, *Psychol. Rev.* **94** (1987) 115–147.
47. J. E. Hummel and I. Biederman, Dynamic binding in a neural network for shape recognition, *Psychol. Rev.* **99** (1992) 480–517.
48. J. M. Anglin, Vocabulary development: A morphological analysis, *Monogr. Soc. Res. Child Dev.* **58** (1993).
49. B. Goldfield and J. S. Reznick, Early lexical acquisition: Rate, content and vocabulary spurt, *J. Child Lang.* **17** (1990) 171–183.
50. G. A. Miller, *The Science of Words* (Freeman New York, 1991).
51. A. L. Woodward, E. M. Markman and C. M. Fitzsimmons, Rapid word learning in 13- and 18-month-olds, *Dev. Psychol.* **30** (1994) 553–566.
52. T. H. Heibeck and E. M. Markman, Word learning in children: An examination of fast mapping, *Child Dev.* **58** (1987) 1021–1034.
53. W. V. O. Quine, *Word and Object* (MIT Press Cambridge, MA, 1960).
54. E. M. Markman, *Categorization and Naming in Children: Problems of Induction* (Cambridge University Press, Cambridge, MA, 1989).
55. P. Gordon, Level-ordering in lexical development, *Cognition* **21** (1986) 73–93.
56. T. R. Haskell, M. C. MacDonald and M. S. Seidenberg, Language learning and innateness: Some implications of compounds research, *Cogn. Psychol.* **47** (2003) 119–163.
57. J. Baxter, Learning internal representations, in *Proc. 8th Int. Conf. Computational Learning Theory*, Santa Cruz, CA, USA (ACM Press, 1995).
58. A. Karmiloff-Smith, *Beyond Modularity: A Developmental Perspective on Cognitive Science* (MIT Press, Cambridge, MA, 1992).
59. R. Pfeifer and C. Scheier, *Understanding Intelligence* (MIT Press, Cambridge, MA, 1999).
60. R. D. Beer, Toward the evolution of dynamical neural networks for minimally cognitive behavior, in *From Animals to Animats: Proc. 4th Int. Conf. Simulation of Adaptive Behavior*, eds. P. Maes, M. Mataric, J.-A. Meyer, J. Pollack and S. W. Wilson (MIT Press, Cambridge, MA, 1996), pp. 421–429.
61. S. Nolfi, Adaptation as a more powerful tool than decomposition and integration, in *Proc. Workshop on Evolutionary Computing and Machine Learning, 13th Int. Conf. Machine Learning*, eds. T. Fogarty and G. Venturini (1996).
62. T. R. Shultz and F. Rivest, Using knowledge to speed learning: A comparison of knowledge-based cascade-correlation and multi-task learning, in *Proc. 7th Int. Conf. Machine Learning* (Morgan Kaufmann, San Francisco, 2000), pp. 871–878.

63. R. French, Semi-distributed representations and catastrophic forgetting in connectionist networks, *Connect. Sci.* **4** (1992) 365–377.
64. T. R. Shultz, F. Dandurand and Y. Takane, Rule following and rule use in simulations of the balance-scale task, submitted.
65. J. A. Fodor, *The Modularity of Mind* (MIT Press, Cambridge, MA, 1983).
66. P. Gallinari, Training of modular neural net systems, in *The Handbook of Brain Theory and Neural Networks*, ed. M. A. Arbib (MIT Press, Cambridge, MA, 1995).
67. D. Gentner, Structure mapping: A theoretical framework for analogy, *Cogn. Sci.* **7** (1983) 155–170.
68. J. Kolodner, *Case-Based Reasoning* (Morgan Kaufmann, San Mateo, 1993).
69. T. R. Shultz, S. P. Mysore and S. R. Quartz, Why let networks grow? in *Neuroconstructivism: Perspectives and Prospects*, Vol. 2, eds. S. S. D. Mareschal and G. Westermann (Oxford University Press, Oxford, 2006), pp. 65–98.
70. J. P. Thivierge, F. Rivest and T. R. Shultz, A dual-phase technique for pruning constructive networks, in *IEEE Int. Joint Conf. Neural Networks* (2003), pp. 559–564.
71. K. Friston, Learning and inference in the brain, *Neural Networks* **16** (2003) 1325–1352.
72. F. G. Ashby, L. A. Alfonso-Reese, A. U. Turken and E. M. Waldron, A neuropsychological theory of multiple systems in category learning, *Psychol. Rev.* **105** (1998) 442–481.
73. M. A. Erickson and J. K. Kruschke, Rules and exemplars in category learning, *J. Exp. Psychol. Gen.* **127** (1998) 107–140.
74. T. J. Palmeri and R. M. Nosofsky, Generalizations by rule models and exemplar models of category learning, in *Proc. 5th Annual Meeting of the Cognitive Science Society* (Erlbaum, Hillsdale, 1993), pp. 794–799.
75. A. Vandierendonck, A parallel rule activation and rule synthesis model for generalization in category learning, *Psychonom. Bull. Rev.* **2** (1995) 442–459.
76. J. P. Thivierge, D. Titone and T. R. Shultz, Simulating frontotemporal pathways involved in lexical ambiguity resolution, in *Proc. 27th Ann. Conf. Cognitive Science Society* (Erlbaum, Mahwah, NJ, 2005), pp. 2178–2183.
77. R. Desimone and J. Duncan, Neural mechanisms of selective visual attention, *Ann. Rev. Neurosci.* **18** (1995) 193–222.
78. E. K. Miller and J. D. Cohen, An integrative theory of prefrontal cortex function, *Ann. Rev. Neurosci.* **24** (2001) 167–202.
79. A. D. Wagner, E. J. Pare-Blagoev, J. Clark and R. A. Poldrack, Recovering meaning: Left prefrontal cortex guides controlled semantic retrieval, *Neuron* **31** (2001) 329–338.
80. S. R. Quartz, Neural networks, nativism and the plausibility of constructivism, *Cognition* **48** (1993) 223–242.
81. D. Mareschal and T. R. Shultz, Generative connectionist networks and constructivist cognitive development, *Cogn. Deve.* **11** (1996) 571–603.
82. J. Fodor, On the impossibility of learning "more powerful" structures, in *The Debate Between Jean Piaget and Noam Chomsky*, ed. M. Piatelli-Palmarini (Routledge & Kegan Paul, 1980), pp. 142–152.
83. M. L. Gick and K. J. Holyoak, Schema induction and analogical transfer, *Cogn. Psychol.* **15** (1983) 1–38.
84. M. L. Gick and K. J. Holyoak, Analogical problem solving, *Cogn. Psychol.* **12** (1980) 306–355.
85. G. Rizzolatti and L. Craighero, The mirror-neuron system, *Ann. Rev. Neurosci.* **27** (2004) 169–192.
86. R. W. Byrne and A. E. Russon, Learning by imitation: A hierarchical approach, *Behav. Brain Sci.* **21** (1998) 667–721.

87. K. Nader, G. E. Schafe and J. E. LeDoux, Fear memories require protein synthesis in the amygdala for reconsolidation after retrieval, *Nature Med.* **406** (2000) 722–726.
88. T. Kohonen, *Self-Organization and Associative Memory*, 2nd edn. (Springer-Verlag New York, 1988).
89. J. Weng and W.-S. Hwang, Online image classification using IHDR, *Int. J. Doc. Anal. Recogn.* **5** (2003) 118–125.
90. R. Sun, P. Slusarz and C. Terry, The interaction of the explicit and the implicit in skill learning: A dual-process approach, *Psychol. Rev.* **112** (2005) 159–192.
91. R. M. Neal, Learning stochastic feedforward networks, Department of Computer Science, University of Toronto (1990).
92. F. Rivest and D. Precup, Combining TD-learning with cascade-correlation networks, in *Proc. 20th Int. Conf. Machine Learning* (AAAI Press, 2003), pp. 632–639.
93. S. Thrun, Towards programming tools for robots that integrate probabilistic computation and learning, in *Proc. IEEE, Int. Conf. Robotics and Automation* (IEEE, San Franciso, 2000).
94. R. Brooks, Elephants don't play chess, *Autonom. Robots* **6** (1990) 3–15.
95. S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 2nd edn. (Prentice Hall, Upper Saddle River, NJ, 2003).



Thomas R. Shultz is Professor of Psychology and Associate Member of the School of Computer Science at McGill University. His Ph.D. is from Yale University in Psychology. Current research interests include connectionism, cognitive science, cognitive development, cognitive consistency phenomena, constraint-satisfaction reasoning, and relations between knowledge and learning. He is a Member of the IEEE Neural Networks Society Autonomous Mental Development Technical Committee and three IEEE Neural Networks Society Task Forces: Reasoning and Inference, Self-organization in Development, and Languages and Language Acquisition.

François Rivest is a Ph.D. candidate in Computer Science and Neuroscience (minor) at Université de Montréal. He has an M.Sc. of Computer Science in Machine Learning from McGill University where his thesis was on the Dean's Honours List. He is also a Student Member of the Society for Neuroscience. His current research interests are models of learning in the brain and computational neuroscience.

László Egri is completing his Master degree in Computer Science in June 2006 and starting his PhD (Computer Science) in September 2007, both at McGill University. He holds a Bachelor degree in Psychology from McGill University. His research interests focus on complexity of constraint satisfaction problems and logical approaches in complexity theory, and it also include compositional learning in neural networks.

Jean-Philippe Thivierge completed his Ph.D. in Psychology at McGill University in 2006, and is currently a Postdoctoral Fellow in the Département de Physiologie at the Université de Montréal in Canada. His research focuses on theoretical neuroscience, with a particular interest in understanding the links between the influence of genetic and neuroanatomical factors on the computational capabilities of neural systems.

Frédéric Dandurand is a professional engineer and currently a Ph.D. candidate in Cognitive Science at McGill University in the Department of Psychology. He got his Bachelor degree in Electrical Engineering from École Polytechnique de Montréal in 1995, and his Master of Engineering degree from McGill University in 1998. He worked as a software engineer in two major computer equipment companies (Matrox and Nortel Networks) from 1998 to 2001. He is interested in cognitive modeling, knowledge representation, learning, and problem solving.