Knowledge-based Cascade-correlation

Thomas R. Shultz
Department of Psychology
McGill University
Montreal, QC Canada H3A 1B1
shultz@psych.mcgill.ca

Francois Rivest
Department of Computer Science
McGill University
Montreal, QC Canada H3A 1B1
frives@po-box.mcgill.ca

Abstract

Neural network modeling typically ignores the role of knowledge in learning by starting from random weights. A new algorithm extends cascade-correlation by recruiting previously learned networks as well as single hidden units. Knowledge-based cascade-correlation (KBCC) finds, adapts, and uses its relevant knowledge to speed learning. In this paper, we describe KBCC and illustrate its performance on a small, but clear problem.

1 Existing knowledge and new learning

Most research on learning in neural networks has assumed that learning is done "from scratch", without the influence of previous knowledge. However, it is clear that when people learn, they make extensive use of their existing knowledge [1-3]. Use of prior knowledge in learning is responsible for the ease and speed with which people are able to learn new material, and for interference effects. A major limitation of neural network models of human cognition and learning is that these networks begin learning from only a random set of connection weights. This implements a *tabula rasa* view of each learning task that few contemporary researchers would accept.

In this paper, we propose a fundamental extension of cascade-correlation (CC), a generative learning algorithm that has been useful in the simulation of cognitive development [4-9]. CC builds its own network topology by recruiting new hidden units into a feed-forward network as needed in order to reduce network error [10]. Our extension, called knowledge-based cascade-correlation (KBCC) recruits previously learned networks in addition to the untrained hidden units recruited by CC. We refer to existing networks as potential *source* knowledge and to a current learning task as a *target*. Previously learned source networks compete with each other and with single hidden units to be recruited into the target network. KBCC is similar to recent neural network research on transfer [11], sequential learning [12], lifelong learning [13], multi-tasking [14], knowledge insertion [15], modularity [16], and input re-coding [17], but it tries to accomplish these functions by storing and searching for knowledge within a unified generative network approach.

2 Description of KBCC

KBCC is similar to CC, except that KBCC treats previously learned networks like single candidate hidden units, in that they are all candidates for recruitment into a target network. A candidate unit and a candidate network both describe a differentiable function. The connection scheme for a sample KBCC network is shown in Figure 1. This scheme is similar to that in CC except that a recruited network can have multiple weighted sums as inputs and multiple outputs, whereas a single recruited unit only has one weighted sum as input and has a single output.

Among the notational conventions we use in formulating KBCC are:

$w_{o_u,o}$: Weight between output o_u of unit u and output unit o.

w_{o_u,i_c} : Weight between output o_u of unit u and input i_c of candidate c.

$f'_{o,p}$: Derivative of the activation function of output unit o with respect to its input at pattern p.

$\nabla_{i_c} f_{o_c,p}$: Partial derivative of candidate c output o_c with respect to its input i_c at pattern p.

$V_{o,p}$: Activation of output unit o at pattern p.

$V_{o_c,p}$: Activation of output o_c of candidate c at pattern p.

$V_{o_u,p}$: Activation of output o_u of unit u at pattern p.

$T_{o,p}$: Target value of output o at pattern p.

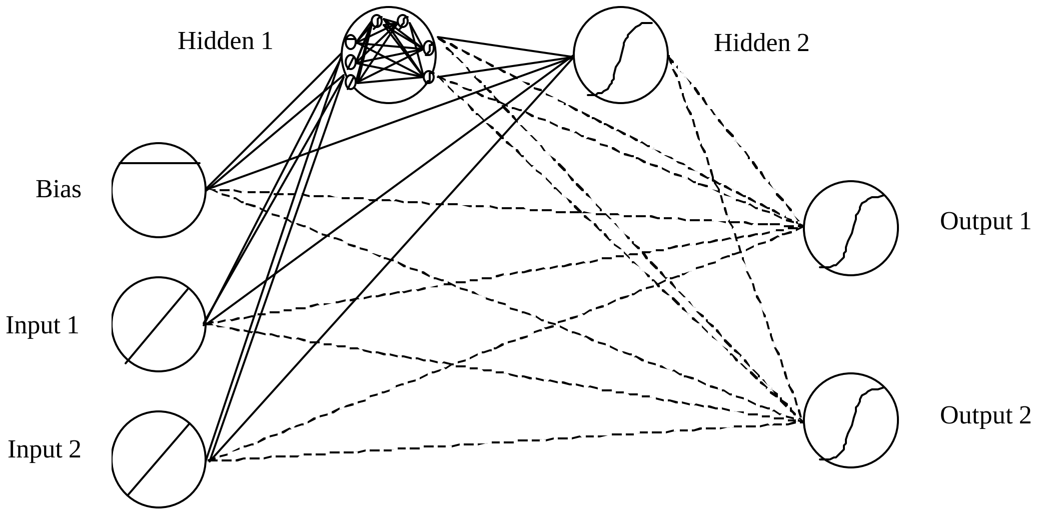

Figure 1: A KBCC network with two hidden units, the first of which is a previously learned KBCC network. The network is shown in an output phase, where dashed lines represent trainable weights and solid lines represent frozen weights. Activation functions are shown inside each unit. The flow of activation is from left to right.

Like CC networks, KBCC networks begin and end in output phase, where weights entering the output units are trained with the Quickprop algorithm [18] in order to reduce error. Weights entering output units are initialized using uniform random numbers within the range of -1 to 1. The function to minimize in the output phase is the sum-squared error over all outputs and all training patterns:

$$F = \sum_{o} \sum_{p} \left(V_{o,p} - T_{o,p} \right)^2 \quad (1)$$

The partial derivative of F with respect to the weight $w_{o_u,o}$ is given by

$$\frac{\partial F}{\partial w_{o_u,o}} = 2 \sum_{p} \left(V_{o,p} - T_{o,p} \right) f'_{o,p} V_{o_u,p} \quad (2)$$

Activation functions for output units are typically sigmoid but can alternatively be linear. An output phase continues until a certain number of epochs pass without solution, error reduction stagnates for a specified few consecutive epochs, or all output activations are within a particular range of their target value. In the last case, learning stops; in the first two cases, there is a shift to input phase.

In the input phase, a new hidden unit is recruited into the network and installed downstream of all existing hidden units. The recruited unit is selected from a pool of candidates. During recruitment, candidates receive input from all existing network units, except output units, and these input weights are trained by trying to maximize a modified correlation between activation of the candidate and network error. Thus, the recruited candidate is the one that is best at tracking the network's current error. Candidates include, not only single units as in CC, but also source networks acquired in past learning. N is the number of candidates per type. Weights entering N single-unit candidates are initialized randomly using a uniform distribution within the range of -1 to 1. Activation functions of the single units are usually sigmoid, but asigmoid and Gaussian functions can also be used. For each previously learned network, input weights for N-1 instances are initialized in the same way. In addition, one instance of each stored network has weights of 1 between corresponding inputs of the target and source networks and 0s elsewhere. The purpose of these identity weights is to enable use of exact knowledge without much additional training.

The function to maximize, again with the Quickprop algorithm [18], is the average covariance of the activation of each candidate c (independently) with the error at each output normalized by the sum-squared error.

$$G_c = \frac{\sum_{o_c}\sum_{o}\left|\sum_{p}\left(V_{o_c,p} - \overline{V}_{o_c}\right)\left(E_{o,p} - \overline{E}_o\right)\right|}{\#O_c \cdot \#O \cdot \sum_{o}\sum_{p} E_{o,p}^2} \tag{3}$$

$\overline{E}_O$ is the mean error at output unit o, and $\overline{V}_{O_C}$ is the mean activation output of candidate C. G_c is standardized by both the number of outputs for the candidate c ($\#O_c$) and the number of outputs for the main network ($\#O$).

The partial derivative of G_c with respect to the weight w_{o_u,i_c} between output o_u of unit u and input i_c of candidate c is given by

$$\frac{\partial G_c}{\partial w_{o_u,i_c}} = \frac{\sum_{o_c}\sum_{o}\sum_{p} s_{o_c,o}\left(E_{o,p} - \overline{E}_o\right)\nabla_{i_c} f_{o_c,p} V_{o_u,p}}{\#O_c \cdot \#O \cdot \sum_{o}\sum_{p} E_{o,p}^2} \tag{4}$$

where $s_{o_c,o}$ is the sign of the covariance between the output o_c of candidate c and the activation of output unit o.

An input phase continues until a certain number of input phase epochs passes without solution, or at least one correlation reaches a minimum value (default value = 0.2) and correlation maximization stagnates for specified few consecutive input phase epochs. When there is a shift to output phase, a set of weights is created from the output(s) of the best candidate to each output unit of the target network. All other candidates are discarded and the newly created weights are initialized using small random values with the sign opposite to that in the correlation.

3 Demonstration of KBCC

We used two kinds of experiments to assess the impact of source knowledge on learning a target task in KBCC. In one experiment, we varied the relevance of the single source of knowledge the network possessed to determine whether KBCC would learn faster if it had source knowledge that was more relevant. In another experiment, we gave networks two sources of knowledge, varying in relevance to a new target problem, to discover whether KBCC would use more relevant source knowledge. Both experiments involved learning whether a pair of Cartesian coordinates were or were not within a particular geometric shape. Source networks varied in terms of translation of the geometric shape.

The input space was a square centered at the origin with sides of length 2. Target outputs specified that the output should be 0.5 if the point described in the input was inside of the figure and -0.5 if the point was outside of the figure. Networks were trained with a set of 225 patterns forming a perfect 15 x 15 grid covering the whole input space including the boundary. There were 200 randomly determined test patterns distributed uniformly over the input space but not used in training.

We ran 20 KBCC networks in each condition of each experiment in order to assess the statistical reliability of results. Learning speed was measured by epochs to learn. Use of relevant knowledge was measured by identifying the source of knowledge that was recruited during input phases.

Knowledge relevance was varied by changing the position of the two-dimensional geometric shape. The target shape in the second phase of knowledge-guided learning was a rectangle sized 0.4 x 1.6 centered at (-4/7, 0) in the input space.

3.1 Effects of single-source knowledge relevance on learning speed

In this experiment, networks had to learn a rectangle (RectL) after having previously learned either a rectangle, two rectangles, or a circle centered at the origin (C), or to the left (L), or right (R) in the input space. The various experimental conditions are shown in Table 1. In a control condition, networks had no source knowledge when beginning the target task, essentially similar to ordinary CC networks. In input phases, the control condition had 8 single-unit candidates; the other conditions each had 4 single-unit candidates and 4 source-network candidates.

Table 1: Single-source Knowledge Conditions

Name	Description	Relation to target
RectL	Rectangle centered at (-4/7, 0)	Exact
2RectLC	2 rectangles, centered at (-4/7, 0) and (0, 0)	Exact/near, overly complex
RectC	Rectangle centered at (0, 0)	Near relevant
RectR	Rectangle centered at (4/7, 0)	Far relevant
2RectCR	2 rectangles, centered at (0, 0) and (4/7, 0)	Near/far, overly complex
Circle	Circle centered at (0, 0) with radius 0.5	Irrelevant
None	No knowledge	No relation

A factorial ANOVA of the epochs to victory yielded a main effect of knowledge condition, $F(6, 133) = 33.15$, $p < .0001$. The mean epochs to victory, with standard deviation bars and homogeneous subsets, based on the LSD post hoc comparison method, are shown in Figure 2. Exact knowledge, whether alone or embedded in an overly complex structure produced the fastest learning, followed by relevant knowledge, distant and overly complex knowledge and irrelevant knowledge, and finally the control condition without any knowledge. Some example output activation diagrams from this simulation are shown in Figure 3.

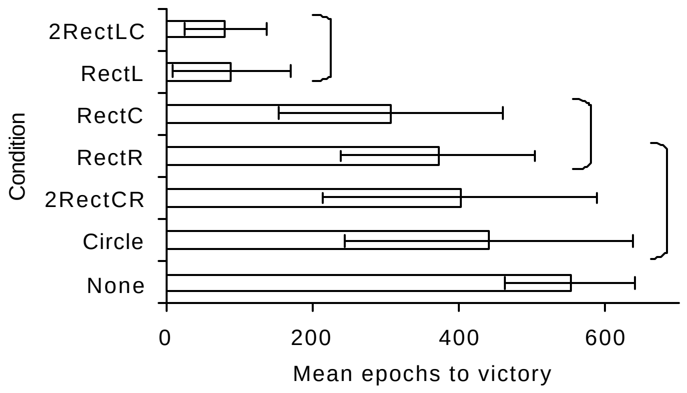

Figure 2: Mean epochs to victory in the target phase, with standard deviation bars and homogeneous subsets.

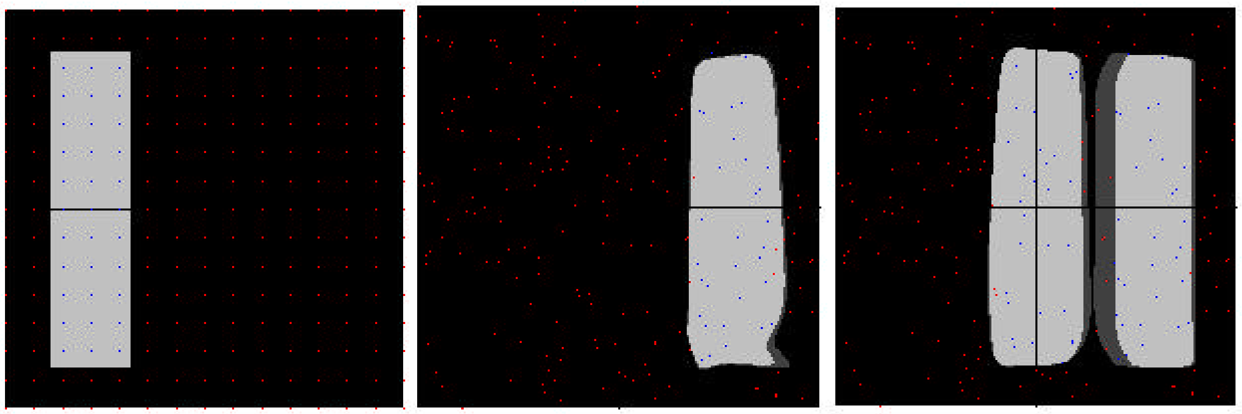

Figure 3: Output activation diagrams showing, from left to right, the target rectangle (RectL), a far but relevant source (RectR), and a far and overly complex source (2RectCR).

3.2 Selection of relevant knowledge from two sources

In this experiment, networks first learned two tasks of varying relevance to the target task of learning RectL. The names of the various knowledge conditions, their relations to the target, and the mean times each network was recruited during input phases are shown in Table 2. Descriptions of the shapes in each condition name were provided in Table 1. In input phases, each target network had 3 single-unit candidates and 3 source-network candidates of each of the two types, for a total of 9 candidates.

Table 2: Dual-source Knowledge Conditions and Mean Networks Recruited

Name	Relation to target	Mean networks recruited			
		RectL	RectR	2RectLC	Circle
RectL, RectR	Exact vs. Relevant	1.0	0.0	n/a	n/a
RectL, 2RectLC	Exact vs. Overly complex	0.95	n/a	0.15	n/a
RectL, Circle	Exact vs. Irrelevant	1.05	n/a	n/a	0.0
RectR, 2RectLC	Relevant vs. Overly complex	n/a	0.15	1.25	n/a
RectR, Circle	Relevant vs. Irrelevant	n/a	1.25	n/a	2.55
2RectLC, Circle	Overly complex vs. Irrelevant	n/a	n/a	1.45	0.15

The two means in each row of Table 2 were compared with a t-test for paired samples (except in the Exact vs. Relevant condition, where there was no variation in either variable). In each case, the mean difference was significant at $p < .001$, $df = 19$. Exact knowledge was always preferred, even when it was embedded in overly complex knowledge. Simple exact knowledge was preferred to overly complex knowledge that had exact knowledge embedded within it. Somewhat surprisingly, knowledge that we had thought would be irrelevant (the circle) was more often recruited than knowledge that we had predicted would be relevant (the rectangle positioned at the right). One reason that so many circle networks were recruited in this condition was that they were not particularly effective in learning the new problem (the rectangle at the left), and thus prolonged learning.

4 Discussion

These results show that KBCC is able to find, adapt, and use its existing knowledge in the learning of a new problem, significantly shortening the learning time. When exact knowledge is present, it is recruited for a quick solution. The more relevant the source knowledge is, the more likely it is recruited for solution of a target problem and the faster that new learning is likely to be. From either an engineering or a cognitive modeling viewpoint, these are the sorts of qualities one would like to see in a system that effectively uses its knowledge in new learning.

In contrast to previous methods for using knowledge in learning, KBCC uses established techniques from generative learning algorithms [10]. Treating its existing networks like untrained single units, KBCC trains weights to the inputs of existing source networks to determine whether their outputs correlate with the target network's error. Network recruitment and integration accomplishes the input re-coding that may convert difficult problems into easier problems [17]. Inputs to a target network are re-coded onto the inputs to a source network in a way that helps to solve the target problem. KBCC trains the output weights from a recruited network in order to incorporate it into a solution of the current problem. This allows KBCC to use knowledge that is only partly relevant to the new task.

Unlike many of the previous techniques for which both the inputs and outputs of the source and target task must match precisely, KBCC can potentially recruit any sort of function to use in a new task. Source network inputs and outputs can be arranged in different orders, employ different coding methods, and exist in different numbers than those in the target network. The wide range of recruitment objects offers more power and flexibility than most knowledge-based learners provide.

KBCC allows for a combination of learning by analogy and/or induction. KBCC learns by analogy to its current knowledge whenever it can and switches to a more inductive mode if it needs to. Recruiting a network is learning by analogy, whereas recruiting a single unit is learning by induction. Both processes are seamlessly integrated in KBCC's approach to a new target task. Finally, KBCC is consistent with the CC algorithm that has been successful in simulating many aspects of cognitive development.

Future work will explore a wider range of applications, including other kinds of shape transformations such as rotation and size changes, large-scale realistic problems, and simulations of psychological experiments on knowledge and learning. We are also examining whether KBCC can improve the quality of learning in cases of impoverished training sets, and whether and under which conditions knowledge interferes with learning.

Acknowledgments

This work was supported by a grant from the Natural Sciences and Engineering Research Council of Canada. We are grateful for comments on an earlier draft from David Buckingham, Jacques Katz, Sylvain Sirois, and Yoshio Takane.

References

[1] Heit, E. (1994) Models of the effects of prior knowledge on category learning. *Journal of Experimental Psychology: Learning, Memory, and Cognition* **20**:1264-1282.

[2] Pazzani, M. J. (1991) Influence of prior knowledge on concept acquisition: Experimental and computational results. *Journal of Experimental Psychology: Learning, Memory, and Cognition* **17**:416-432.

[3] Wisniewski, E. J. (1995) Prior knowledge and functionally relevant features in concept learning. *Journal of Experimental Psychology: Learning, Memory, and Cognition* **21**:449-468.

[4] Buckingham, D., & Shultz, T. R. (1994) A connectionist model of the development of velocity, time, and distance concepts. *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*, pp. 72-77. Hillsdale, NJ: Erlbaum.

[5] Mareschal, D., & Shultz, T. R. (1999) Development of children's seriation: A connectionist approach. *Connection Science* **11**: 149-186

[6] Shultz, T. R. (1998) A computational analysis of conservation. *Developmental Science* **1**:103-126.

[7] Shultz, T. R., Buckingham, D., & Oshima-Takane, Y. (1994) A connectionist model of the learning of personal pronouns in English. In S. J. Hanson, T. Petsche, M. Kearns, & R. L. Rivest (eds.), *Computational learning theory and natural learning systems, Vol. 2: Intersection between theory and experiment*, pp. 347-362. Cambridge, MA: MIT Press.

[8] Shultz, T. R., Mareschal, D., & Schmidt, W. C. (1994) Modeling cognitive development on balance scale phenomena. *Machine Learning* **16**:57-86.

[9] Sirois, S., & Shultz, T. R. (1998) Neural network modeling of developmental effects in discrimination shifts. *Journal of Experimental Child Psychology* **71**:235-274.

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

[11] Pratt, L. Y. (1993) Discriminability-based transfer between neural networks. *Advances in neural information processing systems 5*, pp. 204-211. San Mateo, CA: Morgan Kaufmann.

[12] Silver, D., & Mercer, R. (1996) The parallel transfer of task knowledge using dynamic learning rates based on a measure of relatedness. Connection Science **8**:277-294.

[13] Thrun, S. (1996) *Explanation-based neural network learning -- A lifelong learning approach*. Boston, MA: Kluwer.

[14] Caruana, R. (1992) Multitask learning: A knowledge-based source of inductive bias. *Proceedings of the Tenth International Machine Learning Conference*, pp. 41-48. San Mateo, CA: Morgan Kaufmann.

[15] Shavlik, J. W. (1994). A framework for combining symbolic and neural learning. *Machine Learning* **14**: 321-331.

[16] Jordan, M. I., & Jacobs, R. A. (1994). Hierarchical mixtures of experts and the EM algorithm. *Neural Computation* **6**: 181-214.

[17] Clark, A., & Thornton, C. (1997) Trading spaces: Computation, representation, and the limits of uninformed learning. *Behavioral and Brain Sciences* **20**: 57-97.

[18] Fahlman, S. E. (1988) Faster-learning variations on back-propagation: An empirical study. In D. S. Touretzky, G. E. Hinton, & T. J. Sejnowski (eds.), *Proceedings of the 1988 Connectionist Models Summer School*, pp. 38-51. Los Altos, CA: Morgan Kaufmann.