Using Knowledge to Speed Learning: A Comparison of Knowledge-based Cascade-correlation and Multi-task Learning

Thomas R. Shultz SHULTZ@PSYCH.MCGILL.CA
Department of Psychology, McGill University, Montreal, QC H3A 1B1 Canada

Francois Rivest FRIVES@PO-BOX.MCGILL.CA
Department of Computer Science, McGill University, Montreal, QC H3A 1B1 Canada

Abstract

Cognitive modeling with neural networks unrealistically ignores the role of knowledge in learning by starting from random weights. It is likely that effective use of knowledge by neural networks could significantly speed learning. A new algorithm, knowledge-based cascade-correlation (KBCC), finds and adapts its relevant knowledge in new learning. Comparison to multi-task learning (MTL) reveals that KBCC uses its knowledge more effectively to learn faster.

1. Existing Knowledge and New Learning

Neural networks typically learn *de novo* without the benefit of existing knowledge. However, when people learn, they routinely use their knowledge (Pazzani, 1991; Wisniewski, 1995). Such use of prior knowledge in learning is likely responsible for the ease and speed with which people learn, and for interference with new learning. The technical reason that neural networks fail to use knowledge is that they begin learning from initially random connection weights. This implements a *tabula rasa* view of each distinct learning task that very few cognitive psychologists would accept. In this paper, we compare two algorithms (KBCC and MTL) for their ability to use knowledge to speed learning.

KBCC is an extension of cascade-correlation (CC), a generative learning algorithm often used in the simulation of cognitive development (Buckingham & Shultz, in press; Mareschal & Shultz, 1999; Oshima-Takane, Takane, & Shultz, 1999; Shultz, 1998, 1999; Shultz, Mareschal, & Schmidt, 1994; Sirois & Shultz, 1998). CC constructs its own network topology by recruiting new hidden units into the network as needed in order to reduce error (Fahlman & Lebiere, 1990). KBCC recruits previously learned networks in addition to the single hidden units recruited by CC (Shultz & Rivest, 2000). Following terminology in the literatures on analogy and transfer, 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.

Caruana (1993, 1995, 1997) developed multi-task learning (MTL) in which he trained a network on several tasks taken from the same domain in parallel, with a single output unit for each task. Such networks typically learned a common hidden-unit representation, which produced better generalization than learning the same single tasks one at a time (STL). MTL can be adapted to sequential learning by having a source network generate responses to input values from a new task. These responses can then serve as target output values in parallel MTL of the new task.

This paper reports a comparison of KBCC and MTL on the same sequential learning task. The goals are to determine whether each algorithm can use source knowledge to speed learning and to study of the effects of knowledge relevance on learning speed.

2. Previous Work on Knowledge and Learning

Other previous neural network research on knowledge and learning has included studies of transfer (Pratt, 1993), sequential learning (Silver & Mercer, 1996), lifelong learning (Thrun & Mitchell, 1993), knowledge insertion (Shavlik, 1994), modularity (Jordan & Jacobs, 1994), and input re-coding (Clark & Thornton, 1997).

Pratt (1993) pioneered the study of knowledge and learning in neural networks with a technique called discriminability-based transfer (DBT). DBT uses the weights from a previously trained network to initialize a new network. This seems the most straightforward idea for using knowledge in new neural learning. Because it did not actually work very well, Pratt re-scaled the

previous network's hyper-planes so that useful ones had large weights and less useful ones had small weights.

Silver and Mercer (1996) extended MTL to sequential learning in a method called task rehearsal (TRM). Here, old tasks are pseudo-rehearsed during new learning. In pseudo-rehearsal, a network generates its own target vectors, using its current weights, rather than merely accepting them from the environment (Robins, 1995). In a variation of MTL, separate learning rates for each task are used to control the impact of each source task, ensuring that the most related tasks have the most impact on learning.

Thrun and Mitchell (1993) proposed a technique they called lifelong learning, in which a network meta-learns the slope of the desired function at each training example. This is the derivative of the function at an example output with respect to the input attribute vector. Then, in new learning, a meta-network predicts slopes and estimates its accuracy for each new training example. This technique would seem to trade not so much on knowledge representations as on search knowledge.

Clark and Thornton (1997) emphasized the importance of networks being able to re-code their input in the learning of difficult, so-called Type-2 problems. Type-1 problems are those that can be solved by sampling the originally coded input data. Type-2 problems need re-coding in order to use Type-1 knowledge. Re-coding may require incremental learning, modularity, and representational re-description (Karmiloff-Smith, 1992), but no specific algorithm was proposed.

Shavlik (1994) devised an algorithm for creating knowledge-based artificial neural networks (KBANN). KBANN converts a set of symbolic rules embodying a domain theory of a problem into a feed-forward neural network with the final rule conclusions as output units and intermediate rule conclusions as hidden units. Connection weights and biases are initialized to mimic the conjunctive and disjunctive structures of the original rules. Such knowledge-initialized networks are then trained with examples to refine the network's knowledge. Training with KBANN is typically faster than using standard networks with random weights and leads to better generalization. Following training, the modified rules can be extracted from the network.

Jordon and Jacobs (1994) devised the Hierarchical Mixture of Experts (HME) model to decompose problems into separate network modules. Distinct network modules become expert on subtasks, and cooperate on an overall solution via gating networks that learn to weight the modular expert contributions for particular parts of a problem. HME was found to learn the dynamics of a four-degree-of-freedom robot arm faster than a multi-layer back-propagation network did.

Next we describe in some detail the two learning algorithms featured here: KBCC and MTL.

3. Knowledge-based Cascade-correlation

KBCC learns like CC, except that KBCC treats its previously learned networks as if they were single candidate hidden units. Both single units and existing networks are candidates for recruitment into a target network. A candidate unit and a candidate network each define a function that can be differentiated, which is essential for weight adjustment by gradient descent. The connection scheme for a sample KBCC network is shown in Figure 1. This connection scheme is the same as in CC except that a recruited network can have multiple weighted sums as inputs and can have multiple outputs. In contrast, a single recruited unit, whether in CC or KBCC, has only one weighted sum as input and one output.

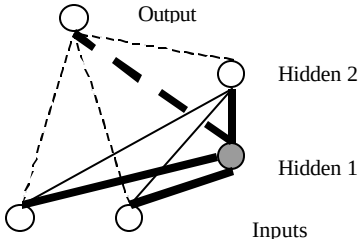

Figure 1. Third output phase of a KBCC network in which the first recruited hidden unit is a previously learned source network with multiple inputs and outputs. Dashed lines indicate trainable weights; solid lines indicate frozen weights. Thin lines indicate single weights; thick lines indicate possible multiple weights to and from the recruited network.

Some notational conventions in our formulation of KBCC:

$w_{o_u,o}$: 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.

KBCC networks begin and end their lives in the so-called output phase, just as CC networks do. In the output phase, weights entering the output units are trained with the quickprop algorithm (Fahlman, 1988) in order to reduce error. Weights entering output units are initialized with uniform random numbers within the range of -1 to 1. The

function to be minimized in the output phase is the sum-squared error over all outputs and all training patterns:

$$F = \sum_o \sum_p \left(V_{o,p} - T_{o,p}\right)^2$$

Like other gradient descent algorithms, KBCC requires computation of the slope of the function to be minimized. The partial derivative of F with respect to the weight $w_{o_u,o}$ is

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

Output units can have either sigmoid or linear activation functions. As in CC, an output phase continues until some number of epochs pass without solution, error reduction stagnates for some few consecutive epochs, or all output activations are within a specified range of their target values. In each of the first two cases, there is a shift to input phase. In the last case, learning stops and the system declares victory.

In an input phase, a new hidden unit is recruited into a network and installed downstream of all existing hidden units. The recruited unit is selected from a pool of candidates. During the recruitment process, candidates receive input from all existing network units, except output units. Input weights are trained by trying to maximize a correlation between activation of the candidate and network error. The candidate that gets recruited is the one that is best at tracking the network's current error.

In KBCC, candidates include, not only single units as in CC, but also previously learned source networks. There are N candidates per type -- single unit and source network. Weights entering the N single-unit candidates are initialized randomly in a uniform distribution within the range of -1 to 1. Activation functions of the single units are typically sigmoid, but can be asigmoid or Gaussian. For each source network, input weights for N-1 instances are initialized in the same way. In addition, one instance of each source network has weights of 1 between corresponding inputs of the target and source networks and 0s elsewhere. These identity weights are designed to enable quick use of exact knowledge.

The function to maximize with quickprop in an input phase is the average covariance of the activation of each candidate c (independently) with the error at each output, normalized by the sum squared error.

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

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

Again, the slope of this function is required for weight adjustment. 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

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

Here, $\sigma_{o_c,o}$ 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 some number of epochs passes without solution, or at least one correlation reaches a minimum value (default value = 0.2) and correlation maximization stagnates for some few consecutive input phase epochs. When there is a shift back to output phase, weights are created from the output(s) of the best candidate to each output unit in the target network. Other candidates are discarded and the new weights are initialized using small random values with sign opposite to that in the correlation.

4. Multi-task Learning

The basic idea underlying MTL is that it can be easier to learn several tasks at once than to learn them separately (Caruana, 1993). MTL has been demonstrated to improve generalization by biasing networks to learn hidden unit representations that are common to several related tasks (Caruana, 1993, 1995, 1997). Baxter (1995) proved that the number of examples required for learning any one task in a multi-task paradigm decreases as a function of total number of tasks learned in parallel.

Caruana (1997) suggested that MTL could be applied to sequential learning by using an existing source network to generate synthetic data that could be added to the training set of a new, related target task. Assuming that the source and target tasks have identical input units and a single output unit, MTL training can then occur in the usual parallel fashion, as shown in Figure 2.

The new training patterns are passed through the source network to generate target output values for the output unit representing the source task in the new MTL network. Because of the requirement that all MTL tasks be learned in parallel, it is not clear whether MTL would inevitably generate faster learning than no knowledge at

all, but Caruana (1995) asserted that "MTB also usually learns in fewer epochs than STB (p. 662)." In this context, MTB refers to multi-task back-propagation and STB refers to single-task back-propagation, more generally called single-task learning (STL).

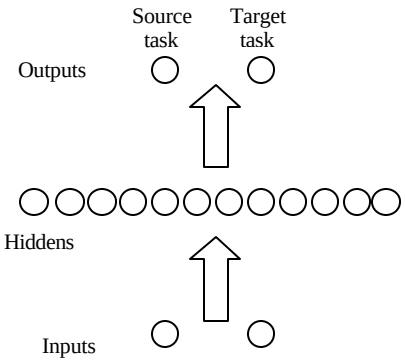

Figure 2. MTL network for two tasks. Arrows indicate full layer-to-layer connectivity.

In any case, Silver and Mercer (1996) applied MTL to sequential learning in this fashion (calling it TRM), and found better generalization and faster learning with MTL than with no knowledge using impoverished training sets, regardless of whether separate learning rates were used for each MTL task. Their problem domain was similar to ours, learning to distinguish the inside vs. the outside of a band in various orientations.

5. Learning Task

To assess the impact of source knowledge on learning a target task, we varied the relevance of a single source of knowledge. The idea was to determine whether KBCC and MTL networks would learn faster if they had source knowledge that was more relevant and to assess the degree of speedup. The task involved learning whether a pair of Cartesian coordinates fell inside or outside of a particular geometric shape. Source networks varied in terms of shape and translation.

The input space was a square centered at the origin with sides of length 2. Networks were trained with a set of 225 patterns forming a regular 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 networks in each condition of each experiment in order to assess the statistical reliability of results. Learning speed was measured by epochs to learn, where an epoch is a pass through all of the training patterns.

Knowledge relevance was varied by changing the position or shape of the source knowledge. 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 (see Figure 3).

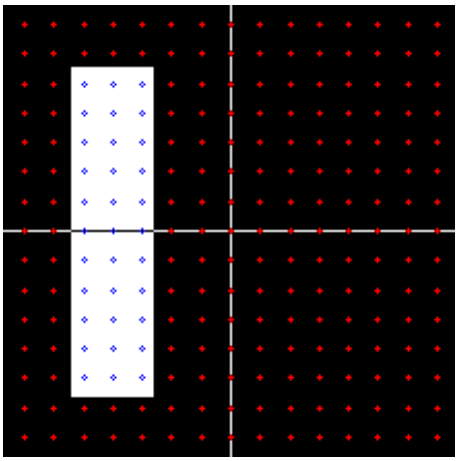

Figure 3. Output activation diagram showing the target problem (left rectangle).

In the target phase, networks had to learn this left-positioned rectangle after having previously learned a rectangle, two rectangles, or a circle centered at the origin, or to the left, or to the right in the input space. The various experimental conditions are shown in Table 1.

Table 1. Source knowledge conditions.

Name	Centered at	Relation to target
Left rectangle	-4/7, 0	Exact
Left & center rectangles	(-4/7, 0) & (0, 0)	Exact/near, overly complex
Center rectangle	0, 0	Near relevant
Right rectangle	4/7, 0	Far relevant
Center & right rectangles	(0, 0) & (4/7, 0)	Near/far, overly complex
Circle	0, 0 with radius 0.5	Irrelevant
None	No knowledge	No relation

Our rectangles are similar to the bands used by Silver and Mercer (1996) except that their bands were not bounded on each end as our rectangles were. Thus, our rectangles might be expected to be more difficult to learn because of their greater non-linearity. In a control condition, networks had no source knowledge when beginning the target task, essentially similar to ordinary CC networks and STL networks. Learning stopped when all output unit activations were within 0.4 of their target values for all

training patterns. Target values for points inside the shape were 0.5; for points outside the shape, -0.5.

5.1 Procedure for KBCC

KBCC networks first learned one of the source problems in Table 1 and then learned the target problem (left rectangle). In the case of the no-knowledge control condition, there was no source to learn so the network just learned the target problem, essentially as a CC network.

In input phases of target learning, the control condition had 8 single-unit candidates; the other conditions each had 4 single-unit candidates and 4 source-network candidates. In any given KBCC network, the source networks were identical except for having four different sets of initially random input weights. This is perfectly analogous to how several different single candidate units are treated in CC.

5.2 Procedure for MTL

For MTL, we used the quickprop algorithm (Fahlman, 1988) because ordinary back-propagation, customarily used in MTL and TRM, could not learn our source tasks within a reasonable time frame. Typically, back-propagation networks ran for over 2000 epochs without making a noticeable drop in error, using any of several different combinations of learning rate and momentum. Quickprop is a weight training algorithm that uses the slope of error with respect to weight at the last two epochs to estimate curvature, the rate at which slope changes as a function of weight change. Thus armed with both slope and an estimate of curvature, quickprop moves more decisively to change weights in order to minimize error than back-propagation, which uses only slope. Moreover, using quickprop to learn weights in MTL ensures a closer comparison to KBCC, which also uses quickprop.

We trained a source network in one of the knowledge conditions in Table 1, and then constructed an MTL network to learn both the source and target tasks in parallel, using outputs from the source network as the target training signal for the source output unit in the MTL network. Training targets for the other output unit in the MTL network were provided by the target task (left rectangle). In the no-knowledge condition, a network was simply trained on the target task in STL fashion.

Twelve hidden units were required to get successful learning of the source and target shapes. Learning rate was 0.5. If source networks failed to learn their shape within 2000 networks, they were discarded and replaced. The stopping criterion was identical for KBCC, STL, and MTL networks -- victory was declared when all output units had activations within 0.4 of their target values for all training patterns.

We next analyze the learning speed results separately for each algorithm to determine whether and to what extent each algorithm benefits from knowledge.

6. KBCC Results

A factorial ANOVA of the epochs to victory in KBCC produced a main effect of knowledge condition, $F(6, 133) = 33.15$, $p < .0001$. The mean epochs to victory, along with standard deviation bars and homogeneous subsets, based on the LSD post hoc comparison method, are shown in Figure 4. The pattern of mean differences reveals that exact knowledge, whether alone or embedded, produced the fastest learning, followed by relevant knowledge, distant and overly complex knowledge and irrelevant knowledge, and finally the control condition without any knowledge. Every knowledge condition was significantly faster than no knowledge at all.

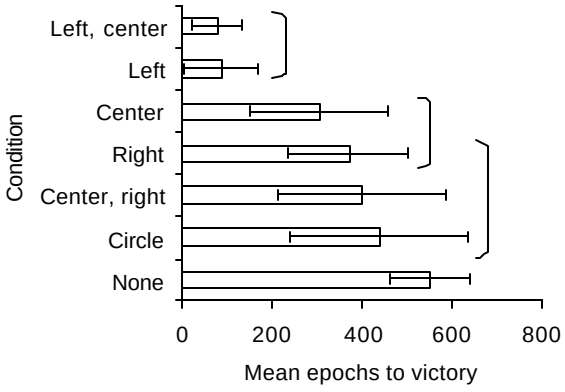

Figure 4. Mean epochs to victory in the target phase for KBCC networks.

Generalization tests with the 200 randomly determined test patterns showed less than 5% misclassification errors in every condition. There were no condition effects on error, indicating that target problems were successfully learned in every condition.

Number of hidden units recruited in the target phase ranged from 3.95 in the center rectangle condition to 7.60 in the center and right rectangle condition, with an overall mean of 5.63. All but two of these recruited units were source networks.

Output activation diagrams were drawn to understand the knowledge representations achieved after source- and target-training phases. Some examples of these diagrams are shown in Figures 5-7. In interpreting these figures, recall that a network learns a shape by distinguishing points within it from points outside of it. In Figures 5-7, white regions of the input space are classified as being inside the shape, black regions outside of the shape, and gray areas are uncertain, meaning that the network gives a borderline, unclassifiable response (in the range -0.1 to 0.1). The horizontal and vertical lines seen in these figures are the *x*- and *y*-axes, respectively, of the input space.

During target learning, the network can recruit single hidden units or already-learned source networks, such as those in Figures 5 and 6. Figure 5 shows a source network's solution to the circle problem. We predicted that the circle would constitute an irrelevant source of knowledge that would not help target learning very much because of the large difference in shape. Figure 6 shows a source network's solution to the left and center rectangles problem. This contains an exact solution to the left rectangle target problem, but is embedded in an overly complex source. Figure 7 shows a KBCC solution to the left rectangle target problem based on recruitment of the network whose knowledge representation is shown in Figure 6. Sources like that in Figure 6 were very effective in speeding up learning, whereas sources like that in Figure 5 were less effective, as reported in Figure 4.

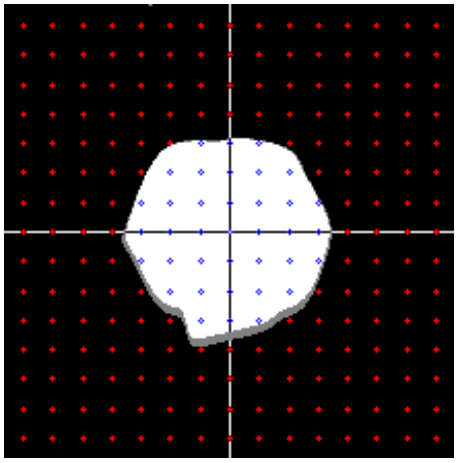

Figure 5. Output activation diagram showing a source network's solution to the circle problem.

The points in each of Figures 5-7 are the 225 target training patterns, which form a 15 x 15 grid covering the whole input space. The learned solutions are irregular because they result from testing the network on a fine grid of 220 x 220 input patterns.

7. MTL Results

Even with quickprop to adjust weights, many networks failed to learn within a reasonable number of epochs in both STL and MTL phases. STL networks were discarded and replaced if they did not learn the source knowledge task within 2000 epochs. There were 3 discarded STL networks in the left rectangle condition, 25 in the center/right condition, 16 in the left/center condition, and 7 in the circle condition. All remaining networks successfully learned their source problem and were thus carried forward to the target phase.

Smaller numbers of MTL networks failed to learn the target problem within 2000 epochs: 1 in the left condition, 1 in the center condition, 5 in the left/center condition, 4 in the center/right condition, and 3 in the no-knowledge condition. Each of these networks was given a score of 2001 epochs.

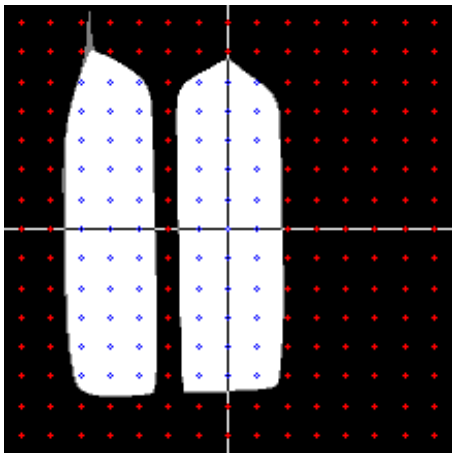

Figure 6. Output activation diagram showing a source network's solution to the left and center rectangles problem.

A factorial ANOVA of the epochs to victory in MTL networks produced a main effect of knowledge condition, $F(6, 133) = 326$, $p < .005$. The mean epochs to victory, along with standard deviation bars and homogeneous subsets, based on the LSD post hoc comparison method, are shown in Figure 8. The pattern of mean differences reveals that the no-knowledge condition did not differ significantly from any knowledge condition, except exact, overly complex knowledge (left and center rectangles). In that comparison, networks without any knowledge learned faster than networks with exact, but overly complex knowledge.

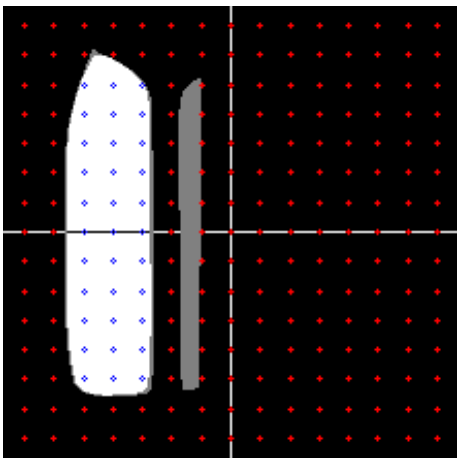

Figure 7. Output activation diagram showing a KBCC solution to the target problem in Figure 3 after recruiting the source problem in Figure 6.

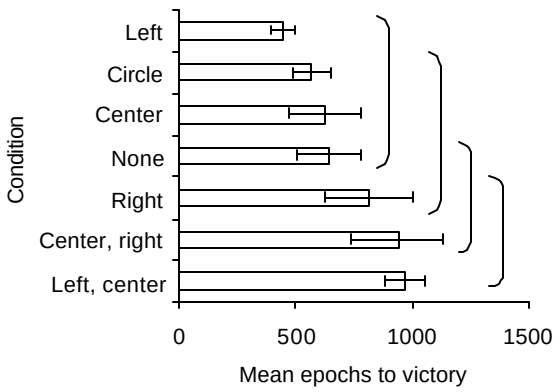

Figure 8. Mean epochs to victory in the target phase for MTL networks.

8. Discussion

These results show that only KBCC networks were able to find and adapt their existing knowledge in new learning, significantly shortening the learning time. When exact knowledge was present, KBCC recruited it for a quick solution. The more relevant the source knowledge, the more likely it was that KBCC recruited it for solution of a target problem and the faster that new learning was likely to be. From any viewpoint, these are desirable properties for a system that effectively uses its knowledge in new learning.

On the other hand, MTL networks did not show any benefits of knowledge for learning speed. They had particular difficulty extracting exact knowledge from an overly complex source network. Moreover, STL networks often failed to learn their assigned source problem and were replaced before MTL could be assessed. MTL may not speed learning of new tasks because it requires both old and new tasks to be freshly learned in parallel. KBCC differs by recruiting, not relearning, old knowledge.

In contrast to all previous methods for using knowledge in learning, KBCC uses established techniques from generative learning (Fahlman & Lebiere, 1990). KBCC treats its existing networks like single-unit candidates, training weights to the inputs of existing source networks to determine whether their outputs correlate with the target network's error.

Network recruitment and integration in KBCC may accomplish the input re-coding needed to convert difficult problems into easier problems (Clark & Thornton, 1997). Inputs to a target network are re-coded onto the inputs to a source network in a way that could help to solve the target problem. In addition, KBCC trains the output weights from a recruited network so as to incorporate the recruited network into a solution of the target problem. Consequently, KBCC is able to use knowledge that is only partly relevant to the target task.

Unlike many of the previous knowledge-based techniques in which both inputs and outputs of the source and target task must match precisely, KBCC can recruit any sort of function to use in a target task. In KBCC, source network inputs and outputs can be arranged in different orders and numbers and use different coding methods than those in the target network. The wide range of recruitment objects offers more power and flexibility than most knowledge-based learners provide. In contrast, MTL requires that the number and ordering of the inputs for each task match precisely and that there is a single output for each task.

KBCC allows for a combination of learning by analogy and induction. It learns by analogy to its current knowledge whenever it can and switches to a more inductive mode if needed. Recruiting a network can be considered as learning by analogy, whereas recruiting a single unit can be regarded as learning by induction. Both kinds of learning are seamlessly integrated as KBCC learns a new target task. KBCC derives its power from the fact that it can learn to use its existing knowledge to solve a target task rather than having to learn the target task from scratch.

In other experiments, we studied what sources KBCC selects when it possesses more than one source of knowledge in its background (Shultz & Rivest, 2000). When present, exact knowledge was always preferred, even when embedded within overly complex knowledge. Simple exact knowledge was preferred to embedded, overly complex knowledge. Occasionally, knowledge that we had predicted to be irrelevant (circle) was recruited more often than knowledge that we had predicted to be relevant though distant (right rectangle). One reason that so many irrelevant sources were recruited was that they were not particularly helpful in learning the new problem (left rectangle), and thus prolonged learning.

There are still significant limitations to KBCC. One is that single hidden units are rarely recruited. Although further study is warranted, our intuition is that humans might be somewhat more likely than this to learn novel tasks from scratch. One solution would be to penalize the complexity of the recruited knowledge in some way.

Another limitation is that search through memory for relevant knowledge rapidly becomes more expensive as knowledge realistically expands. If other, more tractable problems with KBCC can be solved, making memory search more efficient will become a major goal for us.

Current and future work is designed to explore a wider range of applications of KBCC -- to other kinds of shape transformations such as rotation and sizing, psychological simulations of knowledge and learning, and larger-scale realistic problems. We are also testing whether KBCC can improve quality of learning with impoverished training sets, and whether and when knowledge interferes with learning.

Acknowledgements

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, Yoshio Takane, and anonymous reviewers.

References

Baxter, J. (1995). Learning internal representations. *Proceedings of the Eighth International Conference on Computational Learning Theory*. Santa Cruz, CA: ACM Press.

Buckingham, D., & Shultz, T. R. (in press). The developmental course of distance, time, and velocity concepts: A generative connectionist model. *Journal of Cognition and Development*.

Caruana, R. (1993). 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.

Caruana, R. (1995). Learning many related tasks at the same time with backpropagation. *Advances in neural information processing systems 7* (pp. 657-664). Los Altos, CA: Morgan Kaufmann.

Caruana, R. (1997). Multitask learning. *Machine Learning, 28*, 41-75.

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

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.

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.

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

Karmiloff-Smith, A. (1992). *Beyond modularity: A developmental perspective on cognitive science*. Cambridge, MA: MIT Press.

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

Oshima-Takane, Y., Takane, Y., & Shultz, T. R. (1999). The learning of first and second pronouns in English: Network models and analysis. *Journal of Child Language, 26*, 545-575.

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.

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.

Robins, A. V. (1995). Catastrophic forgetting, rehearsal, and pseudorehearsal. *Connection Science, 7*, 123-146.

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

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

Shultz, T. R. (1999). Rule learning by habituation can be simulated in neural networks. *Proceedings of the Twenty-first Annual Conference of the Cognitive Science Society* (pp. 665-670). Hillsdale, NJ: Erlbaum.

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

Shultz, T. R., & Rivest, F. (2000). Knowledge-based cascade-correlation. *Proceedings of the International Joint Conference on Neural Networks*. IEEE Computer Society Press.

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.

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

Thrun, S. & Mitchell, T. (1993). Integrating inductive neural network learning and explanation-based learning. In R. Bajcsy (Ed.), *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*. San Mateo, CA: Morgan Kaufmann.

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