Knowing When to Abandon Unproductive Learning

Thomas R. Shultz (thomas.shultz@mcgill.ca)
Department of Psychology and School of Computer Science, McGill University
1205 Penfield Avenue, Montreal QC, Canada H3A 1B1

Eric Doty (eric.doty@mail.mcgill.ca)
Department of Psychology, McGill University
1205 Penfield Avenue, Montreal QC, Canada H3A 1B1

Frédéric Dandurand (frederic.dandurand@gmail.com)
Department of Psychology, Université de Montréal, 90 ave. Vincent-d'Indy
Montréal, QC H2V 2S9 Canada

Abstract

Autonomous learning is the ability to learn effectively without much external assistance, which is a desirable characteristic in both engineering and computational-modeling. We extend a constructive neural-learning algorithm, sibling-descendant cascade-correlation, to monitor lack of progress in learning in order to autonomously abandon unproductive learning. The extended algorithm simulates results of recent experiments with infants who abandon learning on difficult tasks. It also avoids network overtraining effects in a more realistic manner than conventional use of validation test sets. Some contributions and limitations of constructive neural networks for achieving autonomy in learning are briefly assessed.

Keywords: autonomous learning; abandoning learning; constructive neural networks; SDCC.

Introduction

Autonomous learning is the ability to learn effectively without much external assistance. As such, autonomy is a desired quality in fields such as machine learning and artificial intelligence where the effectiveness of learning systems is seriously compromised whenever human intervention is required. It is likewise a desired feature in cognitive science where a goal is to understand the adaptive functioning of human and other biological agents in their natural environments. An important characteristic of autonomous learners is that they can shape their own learning and development, in large part by choosing what problems to work on. Such choices include selecting a problem to learn and deciding whether to continue learning on the selected task or abandon it in favor of something else.

Knowing When to Quit

Knowing when to stop learning has two obvious components – quitting when the problem has been mastered and when it is unlikely to be mastered. In the constructive neural networks that we favor, victory is declared, and learning terminated, when the network is correct on all training examples, in the sense of producing outputs that are within some score-threshold of their target values (Fahlman & Lebiere, 1990; Shultz, 2003).

Cessation of learning without mastery is considerably more problematic, despite being an important component of autonomous learning in biological agents. It may be useful to analyze such early quitting in terms of costs and benefits. The total cost of learning can be conceptualized as energy expenditure (of the learning effort) plus opportunity cost (the value of the best alternative not chosen, whether other learning or exploitation of resources): $Cost_{Total} = Energy_{Learn} + Cost_{Opportunity}$. Then the net payoff of learning is the benefit of successful learning minus the total cost of learning: $Payoff_{Net} = Benefit_{Learn} - Cost_{Total}$. In continuing to work on an unlearnable problem, there would be a large negative payoff, cost without benefit. Having started to learn such a difficult problem, it could be sensible to abandon it when lack of progress becomes evident.

Previous Work on Abandoning Learning

Recent computational modeling does suggest that a key factor in deciding to abandon learning early is whether learning progress is being made (Schmidhuber, 2005, 2010). In that work, learning progress is monitored by tracking the first derivative of error reduction to identify intrinsic rewards, while a reinforcement-learning module selects actions to maximize future intrinsic rewards. These models curiously conflate novelty with learning success, but it seems more correct to base novelty on initial error, and compute learning success as recent progress in error reduction. These models also include a reinforcement-learning controller that selects actions, and an external network to track learning progress. It seems simpler to continue learning by default until lack of progress is detected, perhaps in terms of stagnation in error reduction.

In an idealized learning model, infant looking was modeled by information-theoretic properties of stimuli (Kidd, Piantadosi, & Aslin, 2010). The negative log probability of an event (corresponding to the number of bits of information conveyed by a stimulus) was conditioned on observing previous events. The larger the negative log probability, the more surprising the current event. As predicted, 7- to 8-month-old infants were more likely to look away from either highly informative or uninformative events. The authors dubbed this the *Goldilocks* effect as

infants prefer to work on tasks that are neither too easy nor too difficult, but just about right in terms of complexity. Although interesting and consistent with an idealized statistical model, these findings are not tied to any neural computational mechanisms. Also, this model is presumably restricted to repeated sequences of events.

Other recent experiments reported that 17-month-olds attend longer to learnable versus unlearnable artificial-language grammars, taking more trials and more time on grammars in which a valid generalization over input utterances could be made (Gerken, Balcomb, & Minton, 2011). Thus, there is now independent evidence that infants may have an implicit metric of their learning progress and can direct their attention to more learnable material.

Constructive Artificial Neural Networks

Constructive artificial neural networks (CANNs) grow a network topology while learning, inspired by principles of brain function and statistical mechanics. Among the attractive features of CANNs are graded knowledge representations, capacity for change and self-organization, and neurological plausibility. CANNs such as cascade-correlation (CC) grow by recruiting new hidden units whose activity correlates with network error (Fahlman & Lebiere, 1990). An extension, sibling-descendant cascade-correlation (SDCC), dynamically decides whether to install a newly recruited unit on the current highest layer (as a sibling) or on its own higher layer (as a descendant), thus optimizing the network topology for the problem being learned (Baluja & Fahlman, 1994). Unit recruitment corresponds roughly to processes of neurogenesis and synaptogenesis in the service of learning (Shultz, Mysore, & Quartz, 2007). Such CANNs have been used to simulate many cognitive, linguistic, and social phenomena while addressing important and longstanding issues about development and learning (Shultz, 2003; Shultz & Fahlman, 2010). They have also yielded testable predictions, many of which have been confirmed in psychological research. Moreover, CANNs have also made considerable progress on several aspects of autonomous learning, including network construction in which new abilities are built on top of earlier achievements.

In the present work, we extend SDCC to abandon learning that is failing to make progress. This is a natural extension for SDCC, which already is able to change phases when it detects lack of progress. Both CC and SDCC operate in two phases: output phase, in which connection weights entering output units are adjusted to reduce network error, and input phase in which weights entering hidden units are adjusted in order to increase the covariance between candidate-unit activation and network error, which ends up recruiting the candidate that best tracks network error. Output phase ends when error reduction stagnates, whereas input phase ends when the covariances between candidate activation and network error stop changing.

We hypothesized that, if error stagnation continues even after recruitment, this could additionally signal that the problem might be unlearnable. This would be the case, for example, on problems with a random structure and insufficient regularities. Of course, some potentially learnable problems are so difficult that their patterns may only seem random. In either case, learning may be frustratingly slow and thus signal to stop and turn to something else more feasible. Here, we apply our extended algorithm to learning problems of varying randomness, discuss its potential to cover the infant experiments just reviewed, and briefly assess the overall ability and limitations of CANNs to learn autonomously.

Method

Algorithm Extension for Abandoning Learning

As noted, each of the two phases in CC and SDCC assesses progress within a phase. We define a learning cycle as an input phase, which recruits a hidden unit, followed by an output phase, which employs the new recruit to help reduce network error. (The first learning cycle has only an output phase, and no input phase.) To assess learning progress across learning cycles, we implemented a new, outside loop to assess progress at the end of each output phase, according to the following algorithm, in which a counter is initialized to 0:

If first learning cycle, then record current error and continue to input phase
Otherwise, compare current error to previous error as absolute difference
  If absolute difference > threshold x previous error, then reset counter to 0 and continue to input phase
  Otherwise,
    If counter = patience, then abandon learning
    Otherwise, increment counter by 1 and continue to input phase

This algorithm is analogous to the progress-assessing loops already used in the output and input phases of CC and SDCC, which compute an absolute difference between a current and previous measure (network error for output-phase and learning-cycle loops, covariance for input-phase loops), and test if this difference is greater than a threshold proportion of the previous value. If the absolute difference exceeds this product, learning continues. If it does not exceed this product, then there is a check to determine if a patience parameter value has been reached. If patience has been exceeded, then the current loop is terminated; otherwise the patience counter is incremented by 1 and learning continues. Resetting the counter to 0 whenever the threshold proportion is exceeded insures that the number of cycles without exceeding the threshold proportion must be consecutive rather than sporadic. Although we rarely alter the threshold and patience parameters for output and input phases, here we do explore some parametric variation for assessing progress across learning cycles.

Continuous XOR

We tested our extended algorithm on a continuous version of the exclusive-or (XOR) problem. This is a well

understood problem in which the simplicity of binary XOR is replaced by a more complex continuous version (Shultz & Elman, 1994; Shultz, Oshima-Takane, & Takane, 1995). Starting from 0.1, input values are incremented in steps of 0.1 up to 1, producing 100 *x*, *y* input pairs that are partitioned into four quadrants of the input space, as illustrated in Figure 1. There is a single output unit with a sigmoid activation function. Values of *x* up to 0.5 combined with values of *y* above 0.5 produce a positive output target (0.5), as do values of *x* above 0.5 combined with values of *y* up to 0.5. Input pairs in the other two quadrants yield a negative output target (-0.5). These constitute the training patterns for conditions that are completely learnable.

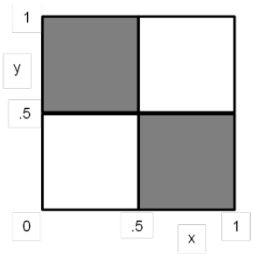

Figure 1. Schematic drawing of the continuous-XOR problem. Gray sectors yield a positive output while white sectors a negative output.

To implement problems of different levels of difficulty, we vary learnability, defined as the percentage of target outputs that are not randomly selected: 0, 25, 50, 75, 80, 85, 90, 95, or 100. If a fresh random number in the range [1, 100] ≥ the particular learnability percentage, then the output target (-0.5 or 0.5) is selected by a .5 chance.

Generalization test patterns are generated by incrementing *x* and *y* values by 0.1 to .94 starting from 0.14. There are 81 such test patterns, all with correct outputs.

In preliminary simulations, it became apparent that learning results were also sensitive to variation in the threshold parameter, so we varied threshold systematically (.05, .1, .15, .2, and .3), while holding patience at 2.

Results

We do not present all of our results here, but only those needed to make important points about basic principles.

Learning Threshold of .15

Typical training-error results are plotted in Figure 2 for two networks, one exposed to patterns with 50% learnability and the other exposed to patterns with 100% learnability. Learning threshold is here set to .15. The diamonds just above the error curves indicate the particular output-phase epochs at which a hidden unit is recruited. As is typical for all threshold values, error is reduced much further with full learnability than with 50% learnability. Moreover, as is typical for thresholds of .1 and higher, learning is abandoned much earlier with 50% learnability than with 100% learnability. These results suggest that the extended algorithm is effective at detecting lack of progress in learning and show what underlies grouped results to follow.

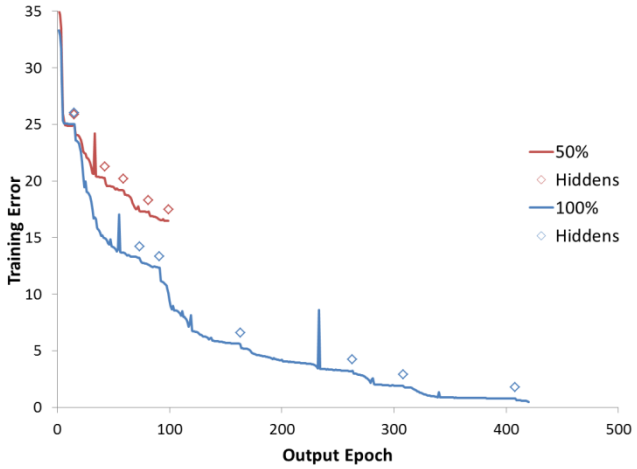

Figure 2. Training error in two networks. With 50% learnability, learning stops at 99 epochs. With 100% learnability, learning stops at 420 epochs.

To see a more general picture, mean per-pattern training error for 20 networks under each learnability condition is plotted in Figure 3, again for a learning threshold of .15. Per-pattern error for a network is computed by dividing total network error by the number of patterns. Each curve is cut off at the mean number of output-phase epochs to abandon learning for that level of learnability, even though some networks surpass this number. Figure 3 provides a more complete demonstration that error reduction is greater with higher learnability and that the extended algorithm is effective at detecting lack of learning progress. Generally, the lower the learnability, the earlier learning is abandoned, at least up to 90% learnability.

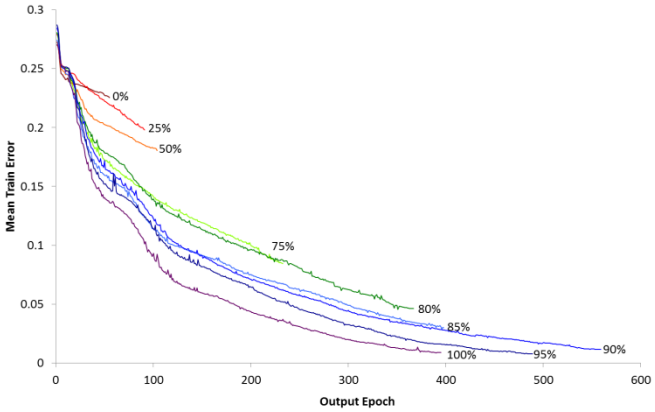

Figure 3. Mean per-pattern training error for 20 networks under each learnability condition over learning cycles.

Mean learning times are shown in Figure 4, which plots the mean output epochs and SE bars for the same 20 networks. This shows more abstractly that low levels of learnability lead to early abandonment of learning.

Moreover, the inverted U-shaped curve reveals a substantial Goldilocks effect wherein networks show more sustained learning for problems of moderate difficulty, peaking at 90% learnability.

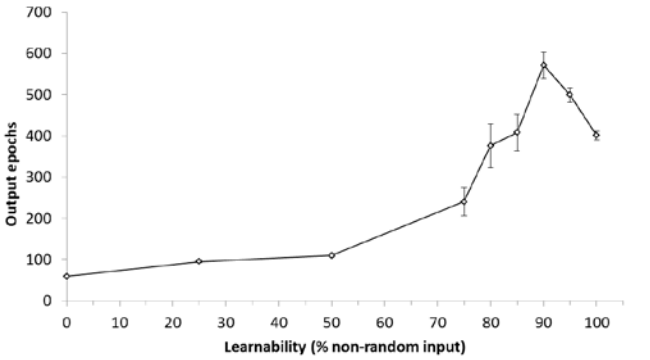

Figure 4. Mean output epochs (with SE bars) for 20 networks with learning threshold of .15.

However, in these simulations, the prolonged learning characteristic of the Goldilocks peak does not often yield superior performance. This is illustrated for these same networks in Figure 5, which plots mean per-pattern test error for each learnability condition. Notice the rise in error on test patterns for learnability conditions in the 50-90% range. Such increases in test error over training suggest that networks are over-fitting the training patterns and starting to memorize the random training patterns instead of abstracting a function to account for the examples. Their earlier success in bringing error down is presumably due to abstracting the continuous-XOR function. But from then on, their only recourse is to start memorizing the random patterns. At 0% learnability, it is impossible to abstract even a basic idea of the exclusive-or problem.

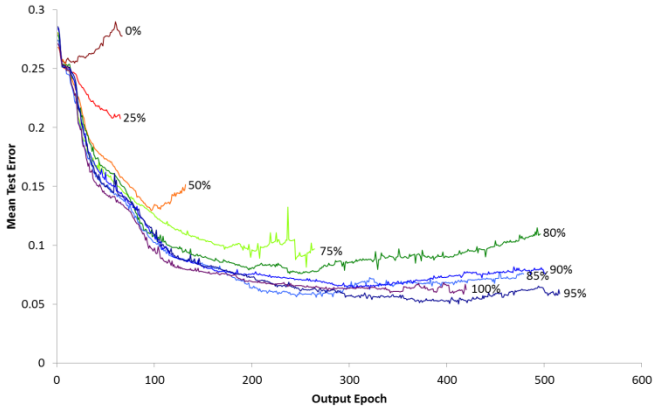

Figure 5. Mean per-pattern test error averaged across 20 networks under each learnability condition.

A rise in test error in what is typically called the validation test set is conventionally used by programmers to determine when to stop network learning. This can be particularly important when using static networks with back-propagation, which have no natural stopping point and where there is no a priori idea of how many hidden units with which to equip a network. Such validation test sets are ordinarily unnecessary for CANNs, which start small and keep growing until the problem is learned. With substantial numbers of random patterns to be memorized, as here, it can be beneficial to also use test error as a training aid, even for CANNs. With a learning threshold of .15, the extended SDCC algorithm was unable to detect, from training error alone, that learning was not progressing, in the sense of generalization ability. Although validation test sets are useful for programmers, they are unrealistic for autonomous learners. Whenever target values and the resulting error signals are available, it is likely that learners would use them to adjust connection weights, thus effectively eliminating such examples from the validation test set.

Learning Threshold of .3

This raises the question of whether other, less sensitive learning-threshold values could be used to curtail learning investment in unproductive tasks like our 50-90% learnability conditions. The answer, as revealed in Figure 6, is *yes* for a learning threshold of .3. In this case, there are no general increases in test error, except at 0% learnability.

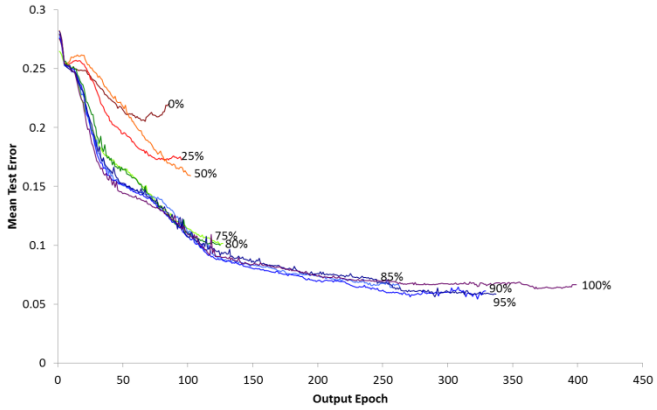

Figure 6. Mean test error averaged across 20 networks under each learnability condition.

However, the Goldilocks effect for these same networks disappears, as revealed in Figure 7. The learning-time peak is now at 100% learnability as all other conditions have abandoned learning earlier. More generally, we find a trade-off between the Goldilocks effect and avoidance of rising test error. As learning threshold increases, the likelihood of finding a Goldilocks effect drops.

Discussion

Interpretation of Results

Our results show that monitoring progress across learning cycles can be used to abandon learning that is unlikely to be successful. This is both realistic and adaptive because, with many problems and domains to learn, it is wasteful to

devote time and energy to learn tasks that are too difficult or impossible. In an abstract sense, on an admittedly different task, our simulations show the ability to capture results like those in two new experiments on learning in human infants. Infants spend more time learning artificial grammars that are possible to learn than they do on grammars that are impossible to learn (Gerken, et al., 2011). Similarly, our neural networks abandon learning impossible tasks, but not tasks that are possible to learn. Further, the network results show that the more difficult the task, the earlier that learning is abandoned, a finding that could serve as a prediction for new human experiments.

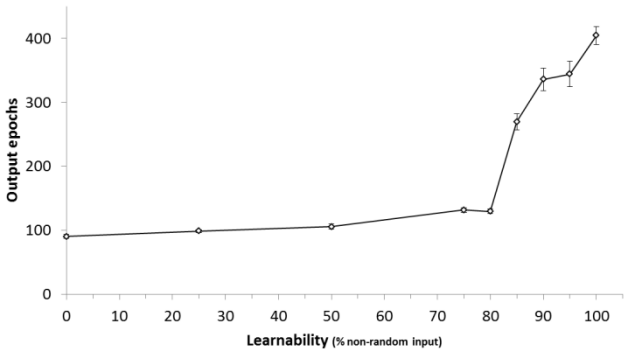

Figure 7. Mean output epochs (with SE bars) for 20 networks, with learning threshold of .30.

Another infant experiment showed a Goldilocks effect in the sense of spending more learning effort on problems of moderate difficulty than on problems that are too easy or too difficult (Kidd, et al., 2010). Our networks show this effect as well, but add a qualification that the Goldilocks effect diminishes at higher levels of a learning-threshold parameter. This offers another prediction to test in human experiments. The psychological equivalent of learning threshold could be sensitivity to changes in error.

In our model, easy tasks are discarded because they have been learned, whereas overly difficult tasks are abandoned because learning has stalled. This identifies two different explanations for turning away from a learning task, one based on success and the other on failure. In contrast, learning may continue as long as some detectable progress is being made.

Our model offers a plausible neural mechanism for such phenomena that allows for further theoretical exploration and extensions. We plan to apply our algorithm to alternate tasks and problems, including those used in psychology experiments and those that vary on dimensions of difficulty other than the proportion of random training patterns.

Our model predicts that learners need to have learning experience with a problem in order to determine whether to continue with it or not. At least with inexperienced learners, there is no shortcut to avoid actually trying to learn. Supporting this idea, we found that amount of first-trial error does not predict learnability on the problems we studied here. Learners may need to give it a serious try before being able to predict whether they might succeed. It would be interesting to see if this is also true of biological learners. If learners exhibit shortcuts to avoid attempted learning, this would imply generalizing across learning content due to previous experience, as when learning shuts down in the presence of mathematical equations.

We also found that overtraining effects can be eliminated with high learning thresholds. This is more realistic for autonomous agents than is monitoring error increases on a validation set of test patterns. Moreover, we find that the Goldilocks and overtraining effects tend to occur in the same circumstances, at relatively low rather than high learning thresholds. Goldilocks peaks are due to the increased learning times caused by low learning thresholds.

There is, of course, more to autonomous learning than abandoning unsuccessful learning. There is also, for example, the choice of which problems to try to learn. We hypothesize that novelty detection, characterized by high initial error, plays a role in choosing learning problems. Abandoning fruitless learning is an essential component of autonomous learning because, as noted, it frees the learner to search for and work on more appropriate problems.

Achieving Autonomy in Learning

Our results show that a small extension to SDCC can provide a useful mechanism for detecting lack of progress in learning, which is an essential component of autonomous learning. In this context, it is worth considering how CANNs such as SDCC fare in terms of other aspects of autonomous learning (Douglas & Sejnowski, 2007). Although there are no completely autonomous artificial learning creatures yet, it is also true that CANNs have made considerable progress in increasing autonomy in learning.

In terms of network construction, SDCC, unlike algorithms for human-designed networks, autonomously designs and builds a network topology that is well suited to the problem being learned. The emerging topology can be flat or deep or anything in between, and learning stops when the problem has been mastered.

Unlike the ordered hierarchies of some static network topologies, SDCC implements a potentially deep, heterarchical topology in which increasingly higher-level, more abstract concepts are composed of simpler ones. Each new hidden unit in SDCC receives signals from input units and any existing lower level hidden units, thus continually building on existing knowledge. Knowledge-representation analysis shows that the first hidden units learn to represent the most obvious and superficial aspects of a problem domain, whereas later hidden units refine and abstract that knowledge (Shultz, 2003). This componential structure is further enhanced in knowledge-based CC (KBCC), where whole, previously learned sub-networks compete to be recruited (Shultz & Rivest, 2001; Shultz, Rivest, Egri, Thivierge, & Dandurand, 2007).

With regard to data selection, like many other artificial neural networks, SDCC focuses on inputs that predict its output, quickly ignoring inputs that are not predictive.

Although such non-predictive inputs are rarely included in practice, it is important to note that, when they are included, they are rapidly and functionally eliminated by learning of near-zero connection weights. It would be feasible to eliminate such detected irrelevant inputs from training patterns altogether, effectively allowing learning to focus attention on what is important while creating a more efficient network.

Among the issues that remain challenges for CAANs, as well as for other network learning algorithms, are single-trial learning, temporal spacing effects, the wake-sleep cycle, synaptic meta-plasticity, relations between brain structure and function, real-time learning in a changing world, and social learning (Douglas & Sejnowski, 2007).

The role of supervision of learning is a complex topic deserving more extended discussion than we can provide here. Suffice it to say that CANNs can learn without a teacher.

For more genuine and more complete autonomy in learning, we believe that it will be important to examine the evolution of learning methods and to implement computational models in robots, with pressures for real-time behavior in fluid environments. Evolution through natural selection is the most plausible natural source of learning mechanisms in both biological and artificial agents (Dunlap & Stephens, 2009). Based on the cost-benefit analysis we presented in the Introduction, it might be possible to show that abandonment of learning itself is favored by natural selection in evolution simulations. And, of course, robotic applications pose a particularly challenging test of learning autonomy.

Acknowledgements

This research was supported by the Natural Sciences and Engineering Research Council of Canada, with an operating grant to TRS and a fellowship to FD. Planning this work benefitted from discussions with LouAnn Gerken, Nick Chater, and Scott Fahlman. We are also grateful to Vincent Berthiaume for relevant pilot work, Simon Reader for pointers to papers on evolution and learning, and Caitlin Mouri for helpful comments on an earlier draft.

References

Baluja, S., & Fahlman, S. E. (1994). Reducing network depth in the cascade-correlation learning architecture. In. Pittsburgh, PA: School of Computer Science, Carnegie Mellon University.

Douglas, R., & Sejnowski, T. (2007). Final workshop report: future challenges for the science and engineering of learning, National Science Foundation (USA). Washington, DC.

Dunlap, A. S., & Stephens, D. W. (2009). Components of change in the evolution of learning and unlearned preference. *Proceedings of the Royal Society B-Biological Sciences, 276*, 3201-3208.

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.

Gerken, L. A., Balcomb, F. K., & Minton, J. L. (2011). Infants avoid 'labouring in vain' by attending more to learnable than unlearnable linguistic patterns. *Developmental Science, 14*, 972-979.

Kidd, C., Piantadosi, S. T., & Aslin, R. N. (2010). The Goldilocks Effect: Infants' preference for stimuli that are neither too predictable nor too surprising. In S. Ohlsson & R. Catrambone (Eds.), *Proceedings of the 32nd Annual Conference of the Cognitive Science Society* (pp. 2476-2481). Austin, TX: Cognitive Science Society.

Schmidhuber, J. (2005). Self-motivated development through rewards for predictor errors / improvements. In *Developmental Robotics 2005 AAAI Spring Symposium*. Stanford University, CA.

Schmidhuber, J. (2010). Formal theory of creativity, fun, and intrinsic motivation. *IEEE Transactions on Autonomous Mental Development (1990-2010), 2*, 230-247.

Shultz, T. R. (2003). *Computational developmental psychology*. Cambridge, MA: MIT Press, (Chapter Chapter).

Shultz, T. R., & Elman, J. L. (1994). Analyzing cross connected networks. In J. D. Cowan, G. Tesauro & J. Alspector (Eds.), *Advances in Neural Information Processing Systems 6* (pp. 1117-1124). San Francisco, CA: Morgan Kaufmannn.

Shultz, T. R., & Fahlman, S. E. (2010). Cascade-Correlation. In C. Sammut & G. I. Webb (Eds.), *Encyclopedia of Machine Learning, Part 4/C* (pp. 139-147). Heidelberg, Germany: Springer-Verlag.

Shultz, T. R., Mysore, S. P., & Quartz, S. R. (2007). Why let networks grow? In D. Mareschal, S. Sirois, G. Westermann & M. H. Johnson (Eds.), *Neuroconstructivism: Perspectives and prospects* (Vol. 2, pp. 65-98). Oxford, UK: Oxford University Press.

Shultz, T. R., Oshima-Takane, Y., & Takane, Y. (1995). Analysis of unstandardized contributions in cross connected networks. In D. Touretzky, G. Tesauro & T. K. Leen (Eds.), *Advances in Neural Information Processing Systems 7* (pp. 601-608). Cambridge, MA: MIT Press.

Shultz, T. R., & Rivest, F. (2001). Knowledge-based cascade-correlation: Using knowledge to speed learning. *Connection Science, 13*, 1-30.

Shultz, T. R., Rivest, F., Egri, L., Thivierge, J.-P., & Dandurand, F. (2007). Could knowledge-based neural learning be useful in developmental robotics? The case of KBCC. *International Journal of Humanoid Robotics, 4*, 245-279.