Learning with Both Adequate Computational Power and Biological Realism

François Rivest
Département d'Informatique et de Recherche Opérationnelle
Université de Montréal
CP 6128 succursale Centre Ville, Montréal, QC H3C 3J7, Canada
francois.rivest@mail.mcgill.ca
and
Thomas R. Shultz
Department of Psychology and School of Computer Science
McGill University
1205 Penfield Avenue, Montreal, QC H3A 1B1, Canada
thomas.shultz@mcgill.ca

Abstract

Computational learning rules considered to be biologically realistic are not only rare but are also known to be seriously underpowered in the sense that they cannot, by themselves, implement the learning that humans and other mammals are capable of. We show mathematically that the computationally-powerful learning rules used in the cascade-correlation family of algorithms can be rewritten in a form that is a small extension of the Hebb rule, which is widely regarded as being biologically realistic. This suggests a way in which computationally-sufficient learning rules could be implemented in real neurons.

Introduction

There is a fundamental dilemma in the computational modeling of the mechanisms of learning, stemming from an apparent incompatibility of biological realism and computational sufficiency. Essentially, the learning rules that are known to be biologically realistic are, by themselves, insufficiently powerful to simulate the learning that humans and other animals are capable of. The purpose of the work reported here is to help resolve this dilemma by showing that some powerful learning rules for artificial neural networks can be rewritten as simple extensions of a learning rule that is considered to be biologically realistic. In doing this, we explicitly lay out the computations in these learning rules in terms of a set of simple, biologically-realistic neural building blocks.

A Biologically-realistic but Computationally-limited Learning Rule

There is one learning rule that most neuroscientists consider to be biologically realistic. This is known as the *Hebb* rule because it was proposed in verbal form by Hebb (1949) in speculations about how learning might occur in the brain. "When an axon of cell A is near enough to cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased" (Hebb, 1949, p. 62).

In neural-network terms, this can be interpreted as strengthening the connection between two units whenever they are simultaneously active. A simple, but more general, formalization of this idea is given in equation 1, which covers both negative and positive activation ranges and also decrements in connection strength. Hebb's verbal formulation dealt only with positive activations and increments in connection strength. Equation 1 says to adjust a weight between two units in proportion to the product of their simultaneous activations x and y, scaled by α, a learning-rate parameter.

$$\Delta w_{i,j} = \alpha x_j y_i \quad (1)$$

Perhaps the first convincing neuroscientific support for Hebbian learning came from a demonstration that electrical stimulation of presynaptic neurons facilitated synaptic transmission (Bliss & Lømø, 1973). This long-term potentiation (LTP) of synaptic efficiency has since been demonstrated in many brain regions of many species. Nearly synchronous firing of pre- and postsynaptic neurons is required for LTP to occur (Kelso, Ganong, & Brown, 1986; Sastry, Goh, & Auyeung., 1986).

The Hebb rule has not fared nearly as well in the computational world. Computational research with artificial neural networks has established that successful Hebbian learning is restricted to learning orthogonal and linearly-separable patterns (McClelland & Rumelhart, 1988). Yet humans and many other species routinely learn non-linearly-separable functions, even those with non-orthogonal patterns (Shultz, 2003). *Exclusive-or* is a simple, well-known problem that is not linearly separable, but is readily learnable. For example, a child may learn that an allowable desert consists of either pie or cake, but not both. As for orthogonality, many ordinary concepts have correlated features, and thus are not composed of orthogonal examples. For example, the feature *has wings* is correlated with the feature *flies* in human concepts of birds. Those examples that possess wings also tend to fly. But these non-orthogonal features would render the *bird* concept impossible for a simple Hebbian rule to learn. The mechanisms by which brains learn non-orthogonal and non-linearly-separable patterns remain unknown.

Computationally Powerful Learning Rules

Some of the more powerful learning rules used in artificial neural networks belong to the cascade-correlation (CC) family. The progenitor is standard CC, which constructs a feed-forward network by recruiting new hidden units as needed, installing each on a distinct layer (Fahlman & Lebiere, 1990). Recurrent CC (RCC) uses recurrent weights on hidden units to process serial stimuli of indefinite length, such as sentences (Fahlman, 1991). Sibling-descendant CC (SDCC) decides whether it is better to install each new hidden unit on the current highest layer (as a sibling) or on its own distinct layer (as a descendant) (Baluja & Fahlman, 1994). Knowledge-based CC (KBCC) evaluates and recruits whole subnetworks as well as single hidden units, thus allowing acquired knowledge to guide new learning (Shultz & Rivest, 2001). Rule-based CC (RBCC) allows for the injection and possible recruitment of symbolic rules within a KBCC framework (Thivierge, Dandurand, & Shultz 2004). Growing and pruning CC (GPCC) prunes away the less useful connections from CC networks, thus improving generalization and interpretability (Thivierge, Rivest, & Shultz 2003). Constraint-satisfaction CC (CSCC) adds a CS network that takes some of its input from the output of a CC network and settles into a stable state, implementing simple inference and reaction time (Shultz & Vogel, 2004).

Like other error-reducing, multi-layered, feed-forward network learning algorithms, those in the CC family easily overcome the computational limitations of the Hebb rule. As long as such networks adjust connection weights in proportion to a product of pre-synaptic activation and a difference between a target and actual post-synaptic activation, they can cope with non-orthogonal patterns. And as long as they contain hidden units with non-linear activation functions, they can learn non-linearly-separable problems (McClelland & Rumelhart, 1988).

Members of the CC family of algorithms have been used to simulate a wide range of psychological phenomena in both cognitive (balance scale, conservation, seriation, concept learning, number comparison, shift learning, and integration of velocity, time, and distance cues) and linguistic (acquisition of personal pronouns, syntax, syllable boundaries, and word stress) domains (Shultz, 2003). These constructive algorithms often prove to be superior to static, feed-forward network algorithms, which adjust connection weights but do not grow, on criteria such as

learning success, learning speed, and coverage of psychological phenomena (Shultz, 2005). The ability to grow and thus build complex knowledge on top of earlier and simpler knowledge appears to be critical to simulation success, particularly for developmental phenomena.

In the rest of this paper, we reformulate the learning rules in CC algorithms in a biologically realistic way using extra units and connections and extended three-way Hebb rules. Our strategy differs from the more usual strategy of trying to make the Hebb rule more powerful. Instead we start with the more powerful learning rules in CC algorithms and make them more Hebbian. At the end of the paper, we consider biological evidence for such extended Hebbian rules.

All CC algorithms alternate between two learning phases: an output phase in which weights entering output units are trained in order to reduce network error, and an input phase in which weights entering candidate recruits are trained in order to correlate with network error.

Output-phase Training in CC Algorithms

The output phase in CC algorithms minimizes the network error described in equation 2, where y_i is activation of the i^{th} output unit, t_i is the corresponding target activation value, and p indexes over training patterns.

$$f = \sum_p \sum_i \left(y_{i,p} - t_{i,p}\right)^2 \tag{2}$$

The derivative of network error with respect to the weights $w_{i,j}$ connecting internal unit x_j to output y_i is given by equation 3.

$$\frac{\partial f}{\partial w_{i,j}} = 2\sum_p \left(y_{i,p} - t_{i,p}\right) y'_{i,p} x_{j,p} \tag{3}$$

Stochastic gradient descent adjusts weights in proportion to the derivative in equation 3. We now reformulate this learning rule in Hebbian terms. Normally CC learns examples with so-called batch learning, in which no connection weights are adjusted until all the training patterns have been processed. This contrasts with per-pattern learning, in which network weights are adjusted after each training pattern is processed. We can transform the batch version of CC into a per-pattern version using a stochastic derivative, which is the same as equation 3, but without the summation.

$$-\frac{1}{2}\frac{\partial f}{\partial w_{i,j}} = -\left(y_{i,p} - t_{i,p}\right) y'_{i,p} x_{j,p} \tag{4}$$

Then we add a new error neuron e_i as shown in Figure 1, which is excited by y_i and inhibited by t_i such that $e_i = y_i - t_i$. If the activation function of y_i is sigmoidal such that $y'_i = y_i\left(1 - y_i\right)$, then the update rule for weight $w_{i,j}$ is given by equation 5, essentially a tri-Hebbian rule.[1]

Equation 5 says that weight change between sending unit *x* and output unit *y* is proportional to the triple product of negative error unit *e*, the derivative of output unit y, and activation of sending

[1] Although it is possible that $y_i\left(1 - y_i\right)$ could be computed by a neuron's internal chemistry, we could alternatively assume another neuron $y_i^- = \left(1 - y_i\right)$ that is inhibited by y_i and that connects to y_i such that $\Delta w_{i,j} = -e_i y_i y_i^- x_j$, making this weight adjustment a quad-Hebbian rule. Still another alternative would be a self-inhibitory link from y_i to itself.

unit x. Activation e_i does not directly affect y_i activation, but the internal chemistry around $w_{i,j}$ modulates change of the weight between *x* and *y*, as specified in equation 5.

$$\Delta w_{i,j} = -e_i y_i' x_j \tag{5}$$

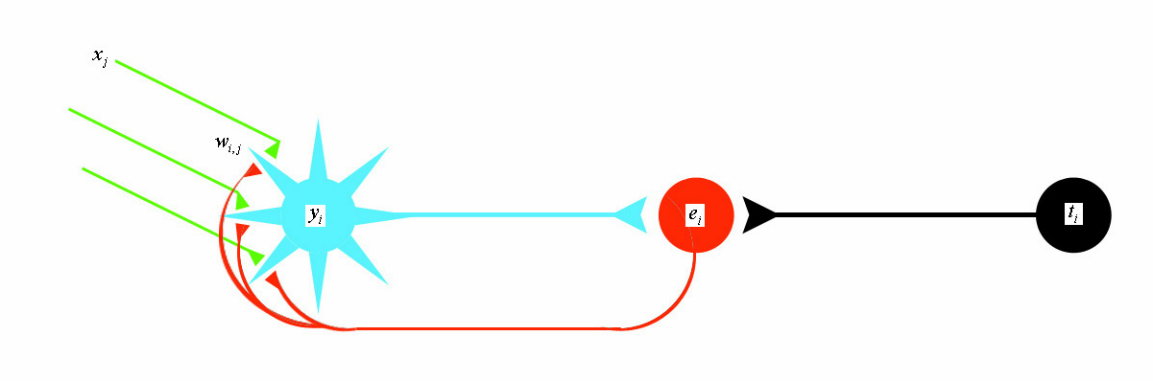

Figure 1: Output-phase neural representation. Notation as in equations 2-5.

Input-phase Training in CC Algorithms

The input phase in CC algorithms maximizes a modified correlation between network error and candidate-unit activation shown in equation 6, where z is a candidate unit,[2] e_i as before, and *p* indexes over patterns. The values $\bar{z}$ and $\bar{e}_i$ are the mean activations of z and e_i over all patterns.

$$g = \sum_i \left| \sum_p (z_p - \bar{z})(e_{i,p} - \bar{e}_i) \right| \tag{6}$$

The derivative of such correlations with respect to weights w_j connecting internal unit x_j to candidate z is given by equation 7, where σ_i is the sign of the correlation between z and e_i.

$$\frac{\partial g}{\partial w_j} = \sum_p \sum_i \sigma_i (e_{i,p} - \bar{e}) z_p' x_{j,p} \tag{7}$$

The first step in creating a Hebbian version of input-phase training is to allow units to build up mean activations of z and e_i. Let $\bar{z}$ be excited by z and itself such that $\bar{z} = \gamma z + (1-\gamma)\bar{z}$ as shown in the top part of Figure 2. Similarly for error units, $\bar{e}_i = \gamma e_i + (1-\gamma)\bar{e}_i$ as shown in the top part of Figure 3. Thus $\bar{z}$ computes the exponential moving average of the z values, and if $\gamma = 1/P$ for P patterns, then after processing a batch of patterns, the value of $\bar{z}$ oscillates around the true (local in time) mean of z.

[2] Because candidate units are trained independently and in parallel, there is no need to describe more than one candidate unit here.

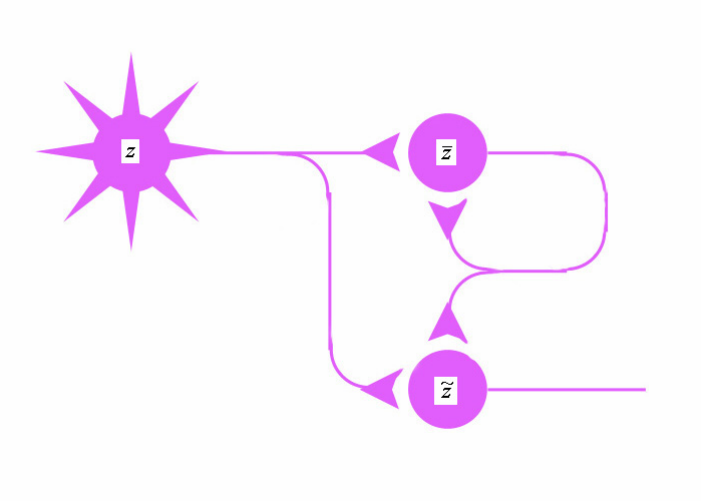

Figure 2: Exponential moving-average unit ($\bar{z}$) and deviation-detector unit ($\tilde{z}$) for candidate-unit z.

Other units are needed to track the deviation terms (e.g., how much each value of z deviates from the mean of z). Let unit $\tilde{z}$ be excited by z and inhibited by $\bar{z}$, such that $\tilde{z} = z - \bar{z}$ as in the lower part of Figure 2. Similarly, $\tilde{e}_i$ is excited by e_i and inhibited by $\bar{e}_i$, such that $\tilde{e}_i = e_i - \bar{e}_i$, as shown in the lower part of Figure 3.

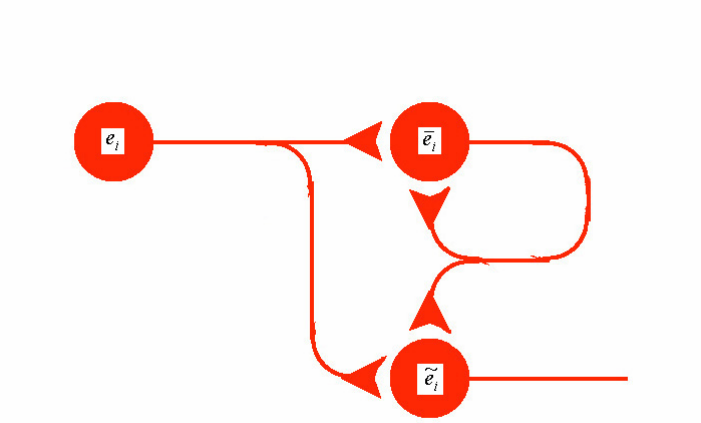

Figure 3: Exponential moving-average unit ($\bar{e}_i$) and deviation-detector unit ($\tilde{e}_i$) for error-unit e_i.

We also need to compute correlations between the deviation terms. Let c_i be a correlation-detector unit for $\tilde{z}$ and $\tilde{e}_i$ such that $c_i = \tilde{z}\tilde{e}_i = (z - \bar{z})(e_i - \bar{e}_i)$, as shown in Figure 4. Then let $\bar{c}_i$ be the exponential moving average of c_i such that after several batches of training, its value oscillates near the true correlation (divided by P). Finally, let unit a be excited by $\tilde{e}_i$ and $\bar{c}_i$ (assuming that $\bar{c}_i$ has a magnitude-independent connection) to get a partial derivative of

correlation $a = \sum_i sign(\bar{c}_i)\tilde{e}_i = \sum_i \sigma_i (e_i - \bar{e}_i)$, as shown in Figure 5. Then the weight-adjustment rule for input-phase training can be described, shown in Figure 6, in tri-Hebbian form in terms of the following modifications of equation 7. As with input-phase training, we transform the batch version of CC into a per-pattern version using a stochastic derivative, which eliminates summation over patterns. Equation 8 indicates that weight change between internal unit *x* and candidate unit *z* is proportional to the triple product of *a* (the partial derivative of the correlation between candidate activation and error), *z'* (the derivative of activation of the candidate unit), and *x* (activation of the sending unit).

$$\frac{\partial g}{\partial w_j} = \sum_i \sigma_i (e_{i,p} - \bar{e}) z'_p x_{j,p}$$

$$\approx a z'_p x_{j,p}$$

$$\Delta w_j = a z' x_j \tag{8}$$

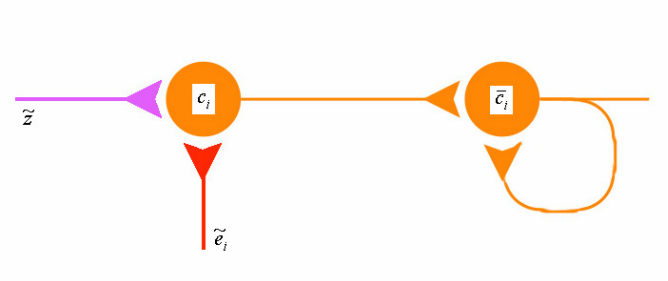

Figure 4: Correlation units track products of candidate-activation deviations and network-error deviations (left), and a moving-average correlation value is computed (right).

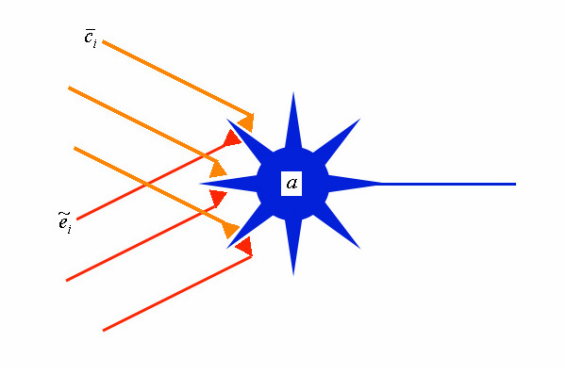

Figure 5: Unit *a* computes the partial derivative of correlation.

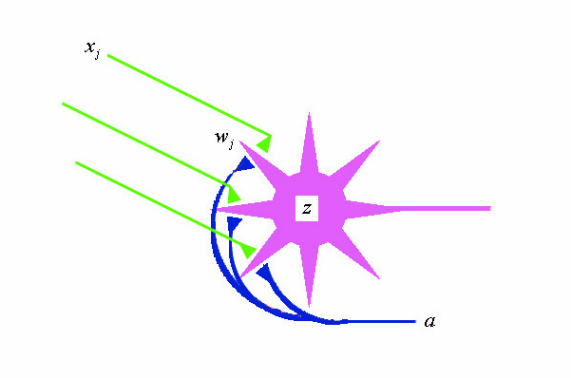

Figure 6: Input-phase neural representation.

Neuroscience Evidence for Three-way Hebbian Rules

Some evidence for three-way Hebbian rules like these comes from studies of the effects of dopamine on synaptic plasticity in corticostriatal circuits that are involved in reinforcement learning in rats. Several experiments have produced results consistent with the view that synapses in these circuits depend on a conjunction of pre- and post-synaptic activity coupled with dopamine-producing neurons (Reynolds & Wickens, 2002; Reynolds, Hyland, & Wickens, 2001). A drawing summarizing such synapses, reproduced in Figure 7, inspired our own circuit diagrams in Figures 1-6.

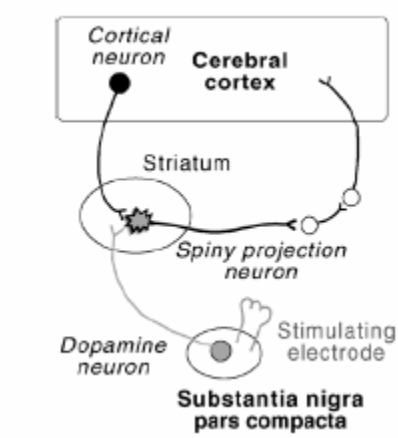

Figure 7: Afferent connections of a striatal spiny projection neuron. From Reynolds and Wickens (2002).

Evidence suggests that induction of Long Term Depression requires a conjunction of presynaptic activity, postsynaptic depolarization, and low levels of dopamine. In contrast, reward-related burst firing of dopamine neurons releases enough dopamine to activate intracellular cascades leading to LTP and reinforcement learning. Importantly, all three parts (presynaptic activity, postsynaptic activity, and amount of activity in dopamine neurons) are required to explain the experimental results. In some experiments, the dopamine neurons were stimulated with an electrode, as shown in Figure 7.

Conclusions

There is wide agreement that the Hebb rule is biologically realistic in terms of its compatibility with known brain functions. However, the Hebb rule alone is not computationally powerful enough to learn the wide range of problems that people and other animals do readily learn. In this paper we show that the mathematics of computationally powerful CC learning algorithms used to simulate a wide range of human learning phenomena can be rewritten in an extended Hebbian form using three units per synapse instead of the usual two units. The main differences between these extended rules and the classic Hebbian rule are the extra units and connection weights that the extended rules require. Because there is neuroscience evidence for three-way Hebbian learning rules, this suggests a way in which the relatively powerful learning rules used in CC algorithms could be implemented in real neurons.

Acknowledgements

This work was supported by a grant from the Natural Sciences and Engineering Research Council of Canada to the second author. Thanks to J.-P. Thivierge for the seminal suggestion that Hebbian learning rules might be able to compute input-phase correlations in CC algorithms and to Marie-Claire Rivest for drawing Figures 1-6. We are grateful to Yoshio Takane, J.-P. Thivierge, and Frederic Dandurand for helpful comments on an earlier draft.

Bibliography

Baluja, S., & Fahlman, S. E. (1994). *Reducing network depth in the cascade-correlation learning architecture*. Technical Report CMU-CS-94-209, School of Computer Science, Carnegie Mellon University.

Bliss, T. V. P., & Lømø, T. (1973). Long-lasting potentiation of synaptic transmission in the dentate area of the anaesthetized rabbit following stimulation of the perforant path. *Journal of Physiology, 232*, 331-356.

Fahlman, S. E. (1991). *The recurrent cascade-correlation architecture*. Technical Report CMU-CS-91-100, School of Computer Science, Carnegie Mellon University.

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.

Hebb, D. O. (1949). *The organization of behavior*. New York: Wiley.

Kelso, S. R., Ganong, A. H., & Brown, T. H. (1986). Hebbian synapses in hippocampus. *Proceedings of the National Academy of Sciences, 83,* 5326-5330.

McClelland, J. L., & Rumelhart, D. E. (1988). *Explorations in parallel distributed processing: A handbook of models, programs, and exercises*. Cambridge, MA: MIT Press.

Reynolds, J. N. J., Hyland, B. I., & Wickens, J. R. (2001). A cellular mechanism of reward-related learning. *Nature, 413*, 67–70.

Reynolds, J. N. J., & Wickens, J. R. (2002). Dopamine-dependent plasticity of corticostriatal synapses. *Neural Networks, 15*, 507-521.

Sastry, B. R., Goh, J. W., & Auyeung, A. (1986). Associative induction of posttetanic and long-term potentiation in CA1 neurons of rat hippocampus. *Science, 232*, 988-990.

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

Shultz, T. R. (2005, in press). Constructive learning in the modeling of psychological development. In Y. Munakata & M. H. Johnson (Eds.), *Processes of change in brain and cognitive development: Attention and performance XXI*. Oxford: Oxford University Press.

Shultz, T. R., & Rivest, F. (2001). Knowledge-based cascade-correlation: Using knowledge to speed learning. *Connection Science, 13*, 43-72.

Shultz, T. R., & Vogel, A. (2004). A connectionist model of the development of transitivity. *Proceedings of the Twenty-sixth Annual Conference of the Cognitive Science Society* (pp. 1243-1248). Mahwah, NJ: Erlbaum.

Thivierge, J.-P., Dandurand, F., & Shultz, T.R. (2004). Transferring domain rules in a constructive network: Introducing RBCC. *Proceedings of the IEEE International Joint Conference on Neural Networks*, 1403-1409.

Thivierge, J.-P., Rivest, F., & Shultz, T. R. (2003). A dual-phase technique for pruning constructive networks. *Proceedings of the IEEE International Joint Conference on Neural Networks 2003*, 559-564.

Wickens, J., & Kotter, R. (1995). Cellular models of reinforcement. In J. C. Houk, J. L. Davis, & D. G. Beiser (Eds.), *Models of information processing in the basal ganglia* (pp. 187-214). Cambridge, MA: MIT Press.