The Evolution of a Generalized Neural Learning Rule

Jeff Orchard
Cheriton School of Computer Science
University of Waterloo
Waterloo, Canada
Email: jorchard@uwaterloo.ca

Lin Wang
Shandong Provincial Key Laboratory of
Network Based Intelligent Computing
University of Jinan
Jinan, 250022, China
Email: ise_wanglin@ujn.edu.cn

***Abstract*—Evolution is extremely creative. The mere availability of a mechanism for synaptic change seems to be enough for evolution to derive a learning rule. Many simulations of evolution have evolved learning in a highly guided manner. Either by constraining the update function to a Hebbian form, or by supplying an error/teaching signal. In this paper, we aim to evolve a more general learning rule. And since neural networks are so versatile, we construct the learning function itself out of a neural network. Our evolved networks excel at the foraging task they evolved in. Amazingly, they even function robustly when tested outside of their historical niche. The same cannot be said for the Hebbian learning networks we compare to.**

I. Introduction

We are interested in the role that synaptic plasticity plays in the evolution of life forms with complex brains. All vertebrates have complex brains that take sensory input (vision, touch, smell, etc.) and determine an appropriate motor output (an action). The brain's function, in essence, is to carry out this mapping. For neural networks (such as brains), the form of the mapping is determined by the strengths of the connections between the neurons. An interesting question is how these connection weights get set. What are the relative roles of genetics and environment.

In this paper, we ask the question: would evolution favour brains that could adjust their own connection weights over brains with static connection weights? And if so, what would the nature of the weight adjustments be?

To answer these questions, we ran a series of simulations that evolved populations of virtual organisms, composed on neural networks. Among the simulations, we evaluated three basic conditions: organisms that have fixed connection weights (denoted *non-learning*), organisms that can adjust their connection weights using a simple, Hebbian plasticity mechanism (denoted *Hebbian update*), and organisms that can adjust their connection weights and biases using a more complex plasticity mechanism based on a neural network (denoted *NWB-update* which stands for "Neural Weights and Biases" update).

II. Related Work

Neural networks have shown to be a flexible method for learning complex mappings from input to output. The challenge is to find the set of neural connection weights that performs the desired transformation on the input to generate the required output. Popular methods such as error backpropagation [1], and contrastive divergence [2] have been used to find connection weights.

Evolutionary algorithms have also been used to optimize the connection weights. Given many input-to-output examples, the fitness function is chosen to reflect the error between the network's output and the desired output. An evolutionary algorithm is used to find connection weights for a neural network that exhibits the same input-to-output behaviour. But those weights are usually fixed, or are changed by a human-designed learning rule such as a Hebbian learning or a delta rule.

The interaction between evolution and learning is more interesting than simply finding fixed connection weights. Neural networks that learn can enhance evolution by smoothing out the fitness landscape [3]. This feedback, called the *Baldwin effect*, arises because learning neural networks can cope with a wider variety of environments. In another study involving foraging, evolving neural networks were given a capacity to learn to predict the outcome of their next move [4]. Even though the networks were not trained to forage better, the networks that learned (or evolved) to predict the outcome of their actions foraged better, and *vice versa*!

But what about evolving the capacity for learning? Can evolution automatically take advantage of a generic synaptic plasticity mechanism to implement a learning rule? The earliest example that we are aware of allowed alteration of each synaptic connection based on a quadratic function of four inputs: pre-synaptic activiy, post-synaptic activity, the current connection weight, and a training signal [5]. They evolved the linear weights for the various terms in the quadratic plasticity function. The resulting neural networks learned to solve a host of linearly-separable binary classification tasks. It demonstrated that evolutionary algorithms can evolve a learning method. However, their network had only an input layer and an output layer, and was strictly feed-forward.

Other examples of evolving learning appeared, many of which used a parameterized Hebbian rule as an update function [6]–[8]. The Hebbian update rule often takes the form

$$\Delta w_{ij} = \eta \left(A x_i x_j + B x_i + C x_j + D \right) ,$$

where x_j and x_i are the pre- and post-synaptic neural activities, Δw_{ij} is the weight adjustment, and A-D and η are parameters. Evolution was tasked with finding the parameter values so that the update rule changed the connection weights in a helpful manner.

A variant of those evolved learning methods involves neuromodulation, in which a separate population of neurons functions to turn on or turn off synaptic plasticity [8], [9]. These methods are effective for implementing reinforcement learning tasks, but – as such – still require a teaching signal, and hence are supervised learning methods.

Some unsupervised evolved learning methods have also been published. For example, Floreano and Urzelai [10] evolved the parameters of a Hebbian update function without an error or teaching signal. A small robot evolved and learned how to efficiently turn on a light and approach it. Unlike most previous implementations of the Hebbian update rule, in which the update function is the same for all connections, Floreano and Urzelai allowed the update rule to vary; each node or connection could choose one of 4 update-rule options, as well as a learning rate. This heterogeneous learning method was encoded in the organism's genome, storing which rule and learning rate was used for each node or connection.

Since then, more heterogeneous update rules have been proposed. One method, dubbed *hyperNEAT*, evolves spatial functions that encode two parameters that control the update function. A more advanced version, called *adaptive ES-hyperNEAT*, stores different Hebbian parameters for each connection using *compositional pattern producing networks* (CPPNs) [11]. These hierarchically-defined functions take the locations of the pre- and post-synaptic neurons as input, as well as the pre- and post-synaptic activity, and generate an update for the connection weight.

Finally, one can even use a **neural network** to compute the update function. Runarsson and Jonsson [12] evolved a neural network that learns to classify binary patterns. Their connection weights are updated throughout each organism's life. Those updates are computed using another neural network. This "learning-rule" network has weights that are fixed within the lifetime of a single individual, but evolved over generations. Their update rule takes, as input, the pre- and post-synaptic activities, as well as a teaching signal. Their networks were very simple – just one feed-forward layer. But they successfully learned to do linear classification.

For a more comprehensive review of evolving plastic networks, see Coleman and Blair [13].

We wanted to break away from the parametric Hebbian paradigm, and continue with Runarsson and Jonsson's idea [12] of using a neural network to compute the update rule. Our hypothesis is that even a 3-layer feed-forward neural network could be used to model a wide array of update functions – far more than the parametric Hebbian update rule. We improve substantially on [12] by adopting a recurrent, fully-connected brain network, and evolve it in a foraging task that requires the organism to adapt.

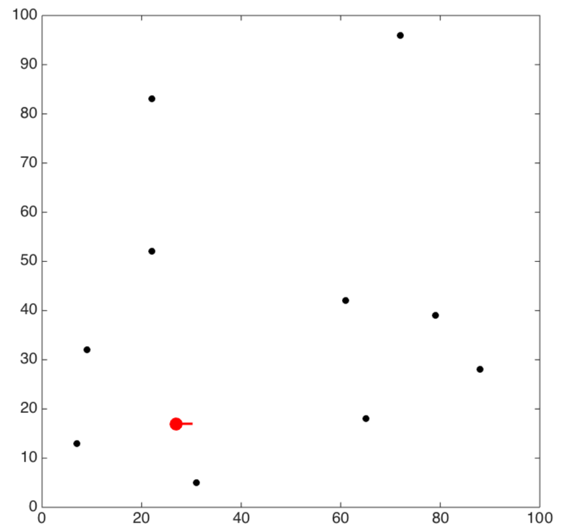

Fig. 1. The foraging arena is 100×100. The dark dots are food items, and the red dot shows the location of the organism, along with an indication of which direction it is facing.

III. Methods

We wanted to see if the simple presence of a complex mechanism for synaptic plasticity was enough to evolve learning. Foraging is a common animal behaviour, so we tested our organisms in a foraging task. Individuals receive information about the location of the nearest food item. The location is input to their neural-network "brain", which then produces output indicating the individual's next action.

We evolved three types of populations: those without synaptic plasticity; those with a parametric Hebbian update rule; and those with an NWB-update rule. We compare the three methods in a number of evolutionary simulations.

Our hypothesis is that populations of neural networks with a facility to update their connection weights will be more able to adapt to varying environments, and be more successful foragers overall.

A. Foraging World

Each organism is placed in a virtual 100×100 arena with 10 food items placed randomly, as shown in Fig. 1. The organism's task is to move around the arena and collect as many food items as possible in a limited amount of time. Based on sensory input, composed of the angle and distance of the nearest food item, the organism has four possible actions: move forward, turn 90° left, turn 90° right, or do nothing. The organism is allowed 50 actions before the food items expire and another set of 10 food items are randomly placed. Each individual is given a 3000-action lifetime during which to collect food.

B. Sensory Offset

To make the task more difficult, we perturb the sensory system of each individual, inducing a lifelong sensory-angle offset. For example, suppose an individual has a sensory offset

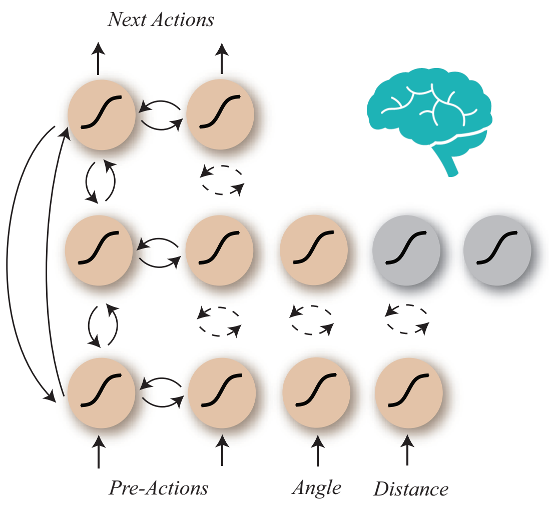

Fig. 2. Neural-network brain. There are 4 input nodes, and 2 output nodes. Different experiments use a different number of internal nodes, either 3 or 5. The network is recurrent and fully connected (bilateral connections between all pairs of neurons).

of 60°. If the nearest food item is at an angle of 30° (at one o'clock), then the sensory input would reflect that the food item is in the direction of 90°.

The fitness of a genome is averaged over 32 individual lifetimes. For each individual, a sensory offset is chosen between $-270°$ and $270°$, and held constant during its lifetime. A genome's fitness is the average number of food items collected in a lifetime, calculated over the 32 individuals with sensory offsets chosen uniformly from $[-270°, 270°]$.

C. Brain Neural Network

Each organism has a "brain", a neural network with four inputs and two outputs. The two output nodes specify the action that the organism will take: (0,0) means do nothing; (0,1) means turn 90° right; (1,0) means turn 90° left; and (1,1) means move forward. Figure 2 illustrates the neural-network brain.

There are four inputs for the neural network, two of which are sensory, and two of which are an efference (internal) copy of the organism's previous action. The two sensory inputs are: distance to the nearest food item, and egocentric angle to the nearest food item. These inputs are the same as those used in [4]. The distance is normalized so that a value of 1 corresponds to 100 grid units (the width of the arena). The angle is also normalized so that a value of 0 corresponds to straight ahead, 0.5 and -0.5 correspond to directly left and right (respectively), and a value of ± 1 indicates the food is in the direction opposite where the organism is facing.

The number of internal nodes (neither input nor output nodes) is either three or five, depending on the experiment (see section IV). These neural networks of either 9 or 11 nodes are fully connected and thus recurrent. That is, each node projects to each other node, and *vice versa*. That means that a network of 9 (11) nodes has 72 (110) internal connections. Including 4 input weights, these brain networks have a total of 76 (114) connection weights.

In addition to the connection weights, each node also has a bias current, adding 9 (11) coefficients to the list of parameters that specify the network.

The network is run in discrete time so that each node's state is overwritten each time step. Furthermore, the updates are synchronous; all nodes have their states updated simultaneously, based on the last time-step's states.

The brain network is used to determine actions from the input (both sensory input, and efference copy of the previous action). The network is run for 10 "deliberation steps" before the state of the output nodes are read to determine the next action. This allows the network to reach a steady state before committing to an action. After each action, the state of each node is reset to a value of 0.5.

D. Neuron Model

Each node in the neural network follows the rule,

$$x_i = f\Big(\beta_i + \sum_j w_{ij} x_j\Big) \tag{1}$$

where x_i is the activity of node i, w_{ij} is the strength of the connection from node j to node i, β_i is the bias current for node i, and $f(\cdot)$ is the logistic function. Hence, each node's activity is in the range $[0,1]$. When each output node is read, its activity is rounded to either 0 or 1 to determine the next action.

E. Synaptic Plasticity

To enable a plastic network (one that can change over its lifetime), we created a mechanism for computing a synaptic-weight update function. Our function is computed by a feed-forward neural network with four input nodes, four internal nodes, and one output node, as illustrated in the left panel of Fig. 3. The network is used to determine an adjustment to each connection in the brain network. For the connection from node j to node i, the update takes the form,

$$\Delta w_{ij} = L\left(w_{ij}\,,\, x_j\,,\, \sum_i w_{ij}\,,\, \beta_i\right)$$

where w_{ij}, x_j, and β_i are as defined earlier, and Σ_i is the total input current impinging on node i from all neurons. The function L represents the update function as computed by the update neural network. Each connection in the brain is updated according to the value Δw_{ij} output by the network, using the rule,

$$w_{ij} \leftarrow w_{ij} + \Delta w_{ij}\,.$$

Since our NWB-update function outputs a value between 0 and 1, we linearly map its value to the range $[-0.005, 0.005]$.

The bias of each node in the brain is also updated by an NWB-update rule. As shown on the right in Fig. 3, an update $\Delta\beta_i$ is generated for each node i.

The nodes in this update-function network follow the same model as shown in (1). However, the weights of the connections are fixed within an individual's lifetime, and arrived at through an evolutionary process.

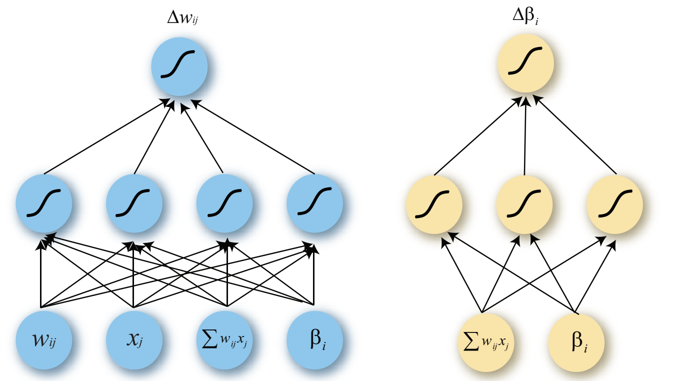

Fig. 3. NWB-update networks: weight update network (left), and bias update network (right). These networks are feed-forward.

F. *Evolutionary Algorithm and Its Parallelization*

Recent studies have shown that artificial evolutionary computation is able to discover models of natural laws [14]. Furthermore, a neural learning rule itself can also be the result of evolution. In this subsection, we describe a genetic algorithm [15] for evolving a learning rule, as well as its corresponding initialization connection weights. The classic binary genetic algorithm is adopted here because it follows the basic theory of evolution without adding other strategies; it enables us to investigate the evolution of learning in a natural way.

The genome consists of two parts: the initial connection weights, and the learning rule. Direct coding is used for the initial connection weights. Each gene represents either an initial weight for a synapse or an initial bias for a neuron. For the learning rule part, it is further divided into two subparts: a global learning rule for synapses, and a global learning rule for biases. Two three-layer feedforward neural networks are used to play the role of these learning rules. Both neural-network-based learning rules are encoded in the genome by direct coding of their connection weights and biases (these networks are evolved, not learned).

In our evolutionary simulations, we evolve populations of 192 genomes (individuals). Each population evolves for 200 generations. A uniform crossover strategy is used with a crossover probability of 0.7. The mutation probability is set to 0.01. Based on the fitness (defined in the next section), the selection operator reproduces the next generation using tournament selection with a size of 15.

Evolutionary algorithms can be very time-consuming. The evolution of an indirect adaptation, like a learning rule, can make the problem even harder. For our experimental platform, we used a CUDA-based high performance computer with a Linux operation system and C programming environment to accelerate the evolution process using parallel computation. The computer consists of four Kepler GK104S GPUs produced by NVIDIA. Each GPU contains 8 stream multiprocessors (SMs), which further contains 192 stream processors (SPs). The clock frequency of each SP is 0.75GHz. Therefore, this computer reaches a peak speed of 9Tflops. This work uses a Master/Slave strategy as the parallelization architecture. It performs the evaluation of individuals using the GPU, and performs the other procedures using the CPU. In total, 6144 threads are performed on GPUs at the same time to evaluate each life time trial for each genome in parallel.

IV. Experiments

We performed three basic simulations in order to compare their performance on the sensory-offset foraging task.

A. *Non-Learning*

As an experimental control condition, we evolved non-learning neural networks to solve the foraging problem. These networks do not have a mechanisms for synaptic plasticity. The connection weights are fixed throughout an individual's lifetime, but the weights evolve over generations. The brains of these organisms have 11 nodes (4 input, 2 output, and 5 internal). Their genome thus contains 125 values encoding the connection weights and biases for the brain.

B. *Hebbian Update Rule*

In a second set of simulations, we implemented a parametric Hebbian update rule. The rule has the form,

$$H(x_i, x_j) = \eta \left(A x_i x_j + B x_i + C x_j + D \right) ,$$

where x_j and x_i are pre- and post-synaptic activities (respectively), and A, B, C, D, and η are parameters to be optimized by evolution. The brains of these organisms have 11 nodes (4 input, 2 output, and 5 internal). Their genomes, therefore, contain a total of 130 values (125 for the brain's initial connection weights and biases, and 5 for the parameters for the Hebbian update rule).

C. *NWB-Update Rule*

In the third set of simulations, we used neural networks to compute the update rule (as described in section III-E). In particular, two networks were used: one to compute connection-weight updates, and the other to compute bias updates.

The connection-weight update network has 4 input nodes (see Fig. 3), 4 internal nodes, and 1 output node (Δw). The bias update network has 2 inputs (the current bias, and input current), 3 internal nodes, and 1 output node ($\Delta \beta$).

To make a fair comparison between the different simulations, we wanted to keep the genome sizes roughly comparable. However, the NWB-update rules require more values to encode their connection weights and biases (25 values for the weight-update network, and 13 values for the bias-update network). To make space in the genome to accommodate these values, we reduced the size of the brain so that it only has 3 internal nodes instead of 5. Hence, the genomes of these organisms contain 85 values for encoding the brain's initial connection weights and biases, and 38 values for encoding the update networks, for a total of 123 values.

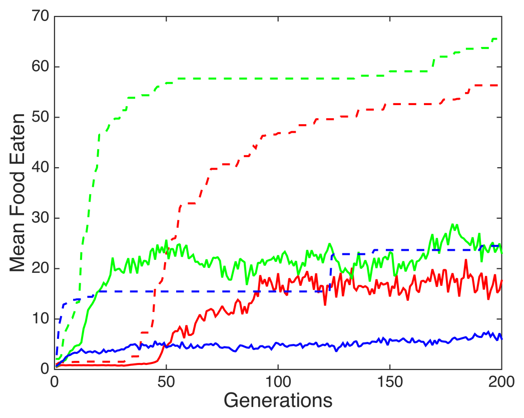

Fig. 4. Evolutionary progression during simulations: non-learning (blue), Hebbian update rule (green), and NWB-update rule (red). The solid lines show the mean number of food items eaten by an individual in a population, while dotted lines show the maximum for each generation.

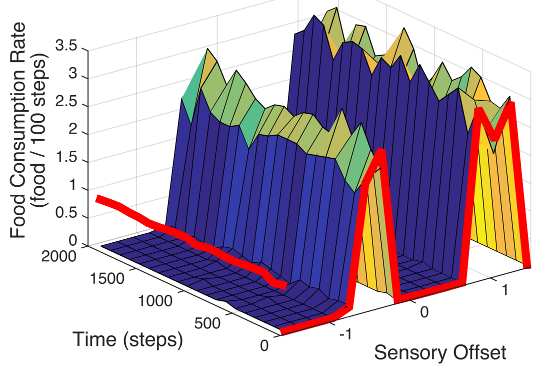

Fig. 5. Lifetime food consumption rate for a non-learning organism, over a range of sensory offsets. For each offset, the lifetime curve represent the average over 20 lifetimes. The red curve on the left gives the consumption rate over the individual's lifetime, averaged over all offsets. The red curve on the right shows the average lifetime consumption rate over various sensory offsets.

D. *Fitness Evaluation*

An organism's fitness naturally depends on its success in locating food. Hence, a large part of the fitness value depends on how many food items it consumes. However, there is also a metabolic cost associated with having a hyper-connected brain with very large connection weights. We included a fitness penalty term for having excessively large connection weights. For the Hebbian simulations, that penalty took the form

$$\left(\sum_{i,j} |w_{ij}| + \sum_{j} |\beta_j| \right) C \,, \tag{2}$$

for some constant C. For the simulations involving the NWB-update function, we used the same weight penalty term, but with a scaled constant C to balance the fact that the smaller brain has fewer connections and fewer biases.

To evaluate how well an organism performs in our foraging world, we simulated it through 32 lifetimes. For each lifetime, we allowed 3000 actions. The 32 lifetimes all had different sensory offsets, chosen uniformly from the range [-1.5, 1.5]. The organism's fitness value is the average number of food items acquired per lifetime, minus the synaptic-size penalty. This ensures that the organisms evolve a general strategy to handle a large range of sensory offsets, while simultaneously bounding the connection weights.

V. Results

A. *Evolutionary Progression*

Figure 4 plots the number of food items eaten during the progress of evolution. The solid lines show each population's mean number of food items eater per lifetime, and the dashed lines show the number of food items eaten by each generation's top individual.

Since the parametric Hebbian update rule has only 5 parameters, it is not surprising that it develops a learning rule more quickly than the NWB-update rule. This can be seen in Fig. 4 as the green curves (Hebbian update) climb more quickly than the red curves (NWB-update).

B. *Robustness to Sensory Offset*

We were surprised how well the non-learning networks performed, despite the varying sensory offsets. However, we discovered that their success was based on expert performance over a small range of offsets, with little or no food consumed for the other (non-preferred) offsets. Figure 5 shows the lifetime food consumption rate for a non-learning individual, over a range of different sensory offsets. For each offset, 20 simulations were run, and the number of food items eaten per 100 steps was averaged over those 20 lifetimes. The Figure shows that this individual is specialized for offsets around -0.5, and 1. For offsets near those values, its performance is exceptionally good. However, half of the offset values yielded poor results. The red line at the left of the figure shows the lifetime food consumption rate averaged over all offsets. We repeatedly observed this behaviour in non-learning evolutionary trials; the resulting best individual showed a preference for small ranges of offsets, but with peaks at different locations.

On the other hand, the organisms that had the NWB-update rule exhibited adaptability to all the sensory offset values we tried. Its performance was the same no matter what the sensory offset was. Figure 6 shows the success of one individual taken from the NWB-update population.

Although not shown here, the Hebbian update rule simulations were similar in nature to the NWB-update rule simulations, showing a brief learning phase at the beginning of life, and exhibiting little or no influence from the sensory offset.

C. *Generalizability*

Figure 7 plots the total number of food items eaten by the top individuals from the non-learning, Hebbian update, and

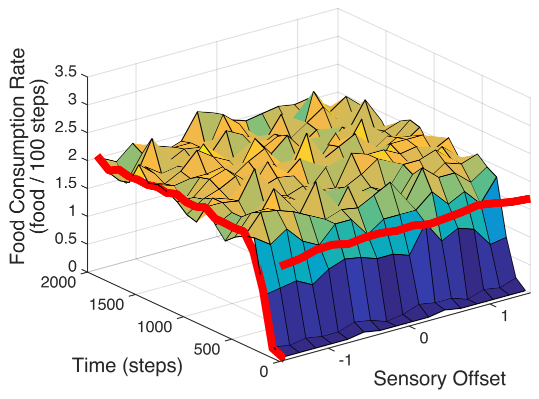

Fig. 6. Lifetime food consumption rate for an organism with an NWB-update rule, over a range of sensory offsets. For each offset, the lifetime curve represent the average over 20 lifetimes. The red curve on the left gives the consumption rate over the individual's lifetime, averaged over all offsets. The red curve on the right shows the average lifetime consumption rate over various sensory offsets.

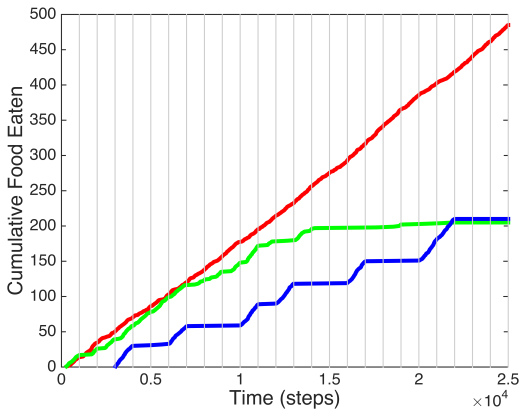

Fig. 7. Total food eaten over an extended lifetime: non-learning (blue), Hebbian update rule (green), and NWB-update rule (red). The sensory offset was changed every 1000 steps to see if the organisms could adapt quickly to changing circumstances.

NWB-update trials over an extended lifetime of 25,000 steps. Recall that these organisms evolved in an environment where a lifetime was 3000 steps, so this extended trial is new to them. Not only that, but the sensory offset was changed every 1,000 steps. During evolution, these organisms did not experience a changing sensory offset within a lifetime.

Not surprisingly, the non-learning individual failed to acquire much food, except when the sensory offset happened to fall within its preferred range. The individual with the Hebbian update rule performed well at first, but then seemed to stall after around13,000 steps. This stalling behaviour was observed with the best individuals from most of the evolutionary trials of the Hebbian update rule.

The individual with the NWB-update rule was able to accommodate all sensory offset changes, and continue collecting food items throughout the entire 25,000-step lifetime.

VI. Discussion

In our experiments, the Hebbian update rule has 5 parameters (spanning 4 degrees of freedom), while the NWB-update rules have a total of 38 parameters. Having more parameters makes the search space much larger, but is it worth it? In our experiments, yes it is. As evolution progresses, the NWB-update populations increase their fitness more gradually, but show more generalizability.

We made the brains of the NWB-update organisms smaller than their non-learning or Hebbian-update counterparts. In particular, the NWB-update organisms had only 3 internal nodes, instead of 5. This was done to equalize the number of variables that the evolutionary search had to optimize over: the non-learning populations had 125 variables, the Hebbian-update populations had 130 variables, and the NWB-update populations had 123 variables. It is not clear how the NWB-update populations would fare if they also had 5 internal nodes, and hence a total of 163 variables. The brain would clearly have more computational power, but the evolutionary search would be in a higher-dimensional space. We plan to investigate this question in the future.

The populations in our study were evolved in a foraging scenario that had a static sensory offset. That is, during evolution, no individual ever had to deal with a **changing** sensory offset. Despite never having been exposed to such hardship, the NWB-update populations were able to adapt to seemingly unlimited changes in sensory offsets during their lifetime. This is in contrast to organisms from the Hebbian-update populations, which seem to eventually saturate their ability to adapt to such changes.

In a previous version of our experiments, the neural learning rule tended to generate unbounded connection weights that diverged linearly over an individual's lifetime. This observation is what prompted us to include the connection weight penalty term in the fitness function, shown in equation (2). Once that penalty term was included, the connection weights of the neural learning organisms stabilized. It is worth noting that the size of the weight penalty term is quite small compared to the number of food items consumed, typically less than 5% for the NWB-update rule organisms, and less than 10% for the Hebbian update rule organisms.

VII. Conclusion

The added complexity of the NWB-update rules was still manageable for our evolution simulations. The NWB-update populations thrived, even in the face of unprecedented sensory updates. After a brief learning phase at the beginning of life, the neural-learning organisms were robust to all sensory offsets that we tested. While their performance was not quite as high as the Hebbian update-rule populations, they were more generalizable.

Many questions still remain. In the future, we would like to investigate the nature of the NWB-update functions. We looked briefly at them, but it will take more careful analysis to decipher how *these* update functions render such robust and flexible brains.

We would also like to study how different metabolic constraints influence the learning function. Will a penalty for neural activity yield more efficient solutions? Will evolving with dynamic sensory offsets induce faster learning? Can the populations still thrive with a greater challenge, such as offsets applied to both sensory inputs?

Acknowledgment

This work was partially supported by the National Natural Science Foundation of China under Grant No. 61573166, No. 61572230, No. 61373054, No. 61472164, No. 81301298, No. 61302128, No. 61472163, Natural Sciences and Engineering Research Council of Canada (NSERC), Shandong Provincial Natural Science Foundation, China, under Grant ZR2015JL025, and the Science and technology project of Shandong Province under Grant No. 2015GGX101025.

References

[1] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[2] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A Fast Learning Algorithm for Deep Belief Nets," *Neural Computation*, vol. 18, pp. 1527–1554, 2006.

[3] G. E. Hinton and S. J. Nowlan, "How learning can guide evolution," *Complex Systems*, vol. 1, pp. 495–502, 1987. [Online]. Available: https://www.complex-systems.com/pdf/01-3-6.pdf

[4] S. Nolfi, J. L. Elman, and D. Parisi, "Learning and Evolution in Neural Networks," *Adaptive Behavior*, vol. 3, no. 1, pp. 5–28, 1994.

[5] D. J. Chalmers, "The evolution of learning: An experiment in genetic connectionism," *Proceedings of the 1990 Connectionist Models Summer School*, pp. 1–20, 1990. [Online]. Available: http://consc.net/papers/evolution.pdf

[6] Y. Niv, D. Joel, I. Meilijson, and E. Ruppin, "Evolution of Reinforcement Learning in Uncertain Environments: A Simple Explanation for Complex Foraging Behaviors," *Adaptive Behavior*, vol. 10, no. 1, pp. 5–24, 2002.

[7] K. O. Stanley, B. D. Bryant, and R. Miikkulainen, "Evolving adaptive neural networks with and without adaptive synapses," *Congress on Evolutionary Computation*, vol. 4, pp. 2557–2564, 2003. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1299410

[8] P. Tonelli and J.-B. Mouret, "On the relationships between synaptic plasticity and generative systems," *Proceedings of the 13th annual conference on Genetic and evolutionary computation - GECCO '11*, pp. 1531–1538, 2011. [Online]. Available: http://portal.acm.org/citation.cfm?doid=2001576.2001782

[9] A. Soltoggio, J. A. Bullinaria, C. Mattiussi, P. Dürr, and D. Floreano, "Evolutionary Advantages of Neuromodulated Plasticity in Dynamic, Reward-based Scenarios," in *Artificial Life XI: Proceedings of the 11th International Conference on Simulation and Synthesis of Living Systems (ALIFE 2008)*. MIT Press, 2008, pp. 569–576. [Online]. Available: https://dspace.lboro.ac.uk/2134/17039

[10] D. Floreano and J. Urzelai, "Evolutionary robots with on-line self-organization and behavioral fitness," *Neural Networks*, vol. 13, pp. 431–443, 2000.

[11] S. Risi and K. O. Stanley, "Indirectly encoding neural plasticity as a pattern of local rules," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6226 LNAI, no. Sab, pp. 533–543, 2010.

[12] T. P. Runarsson and M. T. Jonsson, "Evolution and Design of Distributed Learning Rules," in *IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks*, 2000, pp. 59–63.

[13] O. J. Coleman and S. A. Blair, "Evolving Plastic Neural Networks for Online Learning: Review and Future Directions," in *Artificial Intelligence*, M. Thielscher and D. Zhang, Eds., no. LNCS 7691, 2012, pp. 326–337.

[14] H. L. Michael Schmidt, "Distilling free-form natural laws from experimental data," *Nature*, vol. 324, no. 5923, pp. 81–85, 2009.

[15] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence.*, 2nd ed. Michigan: MIT Press, 1992.