Evolving Plasticity for Autonomous Learning under Changing Environmental Conditions

Yaman, Anil; Mocanu, Decebal Constantin; Iacca, Giovanni; Coler, Matt; Fletcher, George; Pechenizkiy, Mykola

Published in:
ArXiv

Evolving Plasticity for Autonomous Learning under Changing Environmental Conditions

A. Yaman a.yaman@tue.nl
Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, 5612 AP, the Netherlands

D. C. Mocanu d.c.mocanu@tue.nl
Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, 5612 AP, the Netherlands

G. Iacca giovanni.iacca@unitn.it
Department of Information Engineering and Computer Science, University of Trento, Trento, 38122, Italy

M. Coler m.coler@rug.nl
Campus Fryslân, University of Groningen, Leeuwarden, 8911 AE, the Netherlands

G. Fletcher g.h.l.fletcher@tue.nl
Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, 5612 AP, the Netherlands

M. Pechenizkiy m.pechenizkiy@tue.nl
Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, 5612 AP, the Netherlands

Abstract

A fundamental aspect of learning in biological neural networks (BNNs) is the plasticity property which allows them to modify their configurations during their lifetime. Hebbian learning is a biologically plausible mechanism for modeling the plasticity property based on the local activation of neurons. In this work, we employ genetic algorithms to evolve local learning rules, from Hebbian perspective, to produce autonomous learning under changing environmental conditions. Our evolved synaptic plasticity rules are capable of performing synaptic updates in distributed and self-organized fashion, based only on the binary activation states of neurons, and a reinforcement signal received from the environment. We demonstrate the learning and adaptation capabilities of the ANNs modified by the evolved plasticity rules on a foraging task in a continuous learning settings. Our results show that evolved plasticity rules are highly efficient at adapting the ANNs to task under changing environmental conditions.

Keywords

Synaptic plasticity, Continuous learning, evolving plastic networks, evolution of learning, Hebbian learning.

1 Introduction

The evolutionary, developmental and learning levels of organization observed in nature have empowered the abilities of biological organisms to adapt, and stimulated a

broad area of research in nature-inspired hardware and software design (De Castro, 2006; Sipper et al., 1997). In particular, artificial neural networks (ANNs) have proved to be a successful formalization of computational information processing models, inspired by biological neural networks (BNNs) (Rumelhart et al., 1986).

Biologically inspired systems may vary in their levels of complexity. In particular, the conventional models of ANNs are based on the foundation of *connectionism* (Rumelhart et al., 1987); however, their implementation details and learning approaches can differ dramatically.

Inspired by the evolutionary process of biological systems, the research field known as *Neuroevolution* (NE) employs evolutionary computing approaches to optimize ANNs (Floreano et al., 2008; Yao, 1999). Adopting the terminology from biology, a population of individuals are represented as individual *genotypes* consisting of a finite number of *genes* to encode the parameters (i.e. topology, weights and/or the learning approaches) of the ANNs. In other words, individual genotypes can be considered as *blueprints* to construct different ANNs (i.e., corresponding *phenotypes*). Each individual is evaluated on a task, and is assigned a fitness value that measures its performance. A *selection operator* is then used to select better individual —based on their fitness values— that will reproduce (i.e. generate new genotypes) by means of biologically inspired *crossover* and *mutation*. These allow a partially inheritance of genetic material, from parents to offspring, while also introducing variation (Goldberg, 1989). By iterating this cycle over a certain number of *generations*, it is expected to find individuals that are better adapted to the task at hand.

One of the key aspects in NE is the approach used for encoding the ANNs. This, in turn, influences the so-called genotypes-phenotype mapping, i.e. the way a given genotype is used to build a certain phenotype. Broadly speaking, there are two main kinds of encoding approaches: *direct* and *indirect*. In direct encoding, the parameters of the networks (mainly weights and/or topology) are directly encoded into the genotype of the individuals (Yaman et al., 2018; Mocanu et al., 2018); whereas, in indirect encoding, some form of specifications for development and/or training procedures are encoded into the genotype of the individuals (Mouret and Tonelli, 2014; Nolfi et al., 1994). Based on biological evidence, indirect encoding approaches are biologically more plausible in terms of genotype-phenotype mapping, especially considering the large number of neurons and connections in BNNs and the relatively small number of genes in the genotype (Kowaliw et al., 2014). Moreover, BNNs exhibit the *plasticity property* which allows them to change the network configurations during its lifetime and facilitate adaptation under changing environmental conditions.

Among the indirect NE approaches, the evolving plastic artificial neural networks (EPANNs) model the plasticity property of BNNs (Soltoggio et al., 2008; Kowaliw et al., 2014). They encode plasticity rules in the genotype of individuals to specify how configuration of ANNs are adjusted while the network is interacting within an environment. The plasticity property can be limited to the adjustments concerning *synaptic efficiencies (weights)*, and/or can also include changes on the network's *structure/topology* (i.e. formation/elimination of new/existing connections) (Mouret and Tonelli, 2014; Nolfi et al., 1994). Since most of the changes are observed during the lifetime of the networks, the same genetic code can produce entirely distinct network configurations.

Hebb (1949) proposed a biologically plausible plasticity mechanism known as *Hebbian learning*, which performs synaptic adjustments based on the local activation of neurons. According to Hebbian learning, the synaptic efficiencies between pre- and post-synaptic neurons are strengthened/weakened if the neurons' activation states are

positively/negatively correlated. However, the basic formalization of Hebbian learning can suffer from instability as it introduces an indefinite increase/decrease of the synaptic efficiencies (Vasilkoski et al., 2011). Several other modified variants have been proposed to reduce this effect (Brown et al., 1990; Vasilkoski et al., 2011; Sejnowski and Tesauro, 1989). Nevertheless, these plasticity rules may still require further optimization to properly capture the dynamics needed for adjusting the network parameters.

A number of previous works proposed optimizing the parameters of some form of Hebbian plasticity rules using evolutionary approaches (Floreano and Urzelai, 2000; Niv et al., 2002; Soltoggio et al., 2008). However, these attempts may involve a high degree of complexity that may not be able to deliver insights into the dynamics of the plasticity property (Orchard and Wang, 2016; Risi and Stanley, 2010). For instance, some of the existing works evolve the initial synaptic weights and/or the connectivity of the networks in addition to the plasticity rule. However, evolving the initial synaptic weights of the networks increases the number of parameters to evolve and, in principle, can be decoupled from the evolution of plasticity rules (except for tasks where there is a need to adapt to changing conditions); on the other hand, evolving the connectivity of the networks may overfit the networks to a certain task, making it difficult to evolve adaptive behavior.

In this work, we propose a novel approach to produce plasticity property in ANNs for a continuous learning scenario with changing environmental conditions. We employ Genetic Algorithms to evolve discrete plasticity rules to determine how synaptic weights are adjusted based on the activation states of the connected neurons, and a reinforcement signal received from the environment. One of the main advantages of our approach is that the plasticity rules it evolves can provide interpretable results, as an alternative to the other plasticity rules proposed in the literature.

We use relatively large networks consisting of one hidden layer, and introduce local weight competition to allow self-organized adaptation of synaptic weights. We demonstrate the lifetime learning and adaptation capabilities of plastic ANNs on a foraging task where an agent is required to learn to navigate within an enclosed environment, and collect/avoid specific types of items. Starting from the randomly initialized values, weights are updated after each action step, based on the reinforcement signals received from the environment. We test the adaptation capabilities of the networks by switching the types of items to be collected/avoided after a certain number of action steps. We show that the evolved plasticity rules are capable of producing stable learning, and autonomous adaptation capabilities under changing environmental conditions. This form of learning can be seen as a distributed, self-organized continuous learning process that can be carried on without the need for global evaluation or validation.

The rest of the paper is organized as follows: in Section 2, we provide the background for Hebbian learning, and discuss the literature on the evolution of learning. In Section 3, we introduce our approach to evolving plasticity rules and provide the details of the genetic algorithm. In Section 4, we present the experimental setup. In Section 5, we provide the results for the learning and adaptation capabilities of the networks modified by the evolved plasticity rules; finally, in Section 6 we conclude by recapitulating our main results and highlighting possible future works.

2 Background

ANNs consist of a number of *artificial neurons* arranged within a certain connectivity pattern, where a directed connection between two neurons is referred as a *synapse*, a

neuron with an outgoing synapse is referred to as a *pre-synaptic neuron*, and a neuron with an incoming synapse is referred to as a *post-synaptic neuron*. Various types of ANNs have been designed by arranging neurons according to different topologies (De Castro, 2006). For instance, an example of a *fully connected Feed Forward ANN* is provided in Figured 4.

The activation of a post-synaptic neuron, a_i, is typically calculated as:

$$a_i = \psi\left(\sum_{j=0} w_{i,j} \cdot a_j\right) \tag{1}$$

where a_j is the activation of the j-th pre-synaptic neuron, $w_{i,j}$ is the synaptic efficiency (weight) from from pre-synaptic neuron j to post-synaptic neuron i, a_0 is the bias which usually takes a constant value of 1, and $\psi()$ is the activation function.

2.1 Hebbian learning

In its general form, Hebbian learning rule is formalized as:

$$w_{i,j}(t+1) = w_{i,j}(t) + m(t) \cdot \Delta w_{i,j} \tag{2}$$

where the synaptic efficiency $w_{i,j}$ at time $t+1$ is updated by the change $\Delta w_{i,j}$ that is a function of a_i and a_j:

$$\Delta w_{i,j} = f(a_i, a_j) \tag{3}$$

A modulatory signal, $m(t)$, is used to determine the sign of the Hebbian learning. If $m(t)$ is positive and there is a positive correlation between the activations of neurons, the synaptic efficiency between them is strengthened; whereas if their activations are not correlated then the synaptic efficiency between them is weakened. The negative sign of $m(t)$ reverses to sign of the Hebbian learning (which in this case is also known as *anti-Hebbian learning*), by strengthening the synaptic efficiencies between neurons with uncorrelated activations, and weakening the synaptic efficiencies of neurons with correlated activations. The modulatory signal is usually equivalent to the reward received from the environment, unless other kinds of modulatory signaling mechanisms are used, such as neuromodulation (Soltoggio et al., 2008).

A "plain" Hebbian rule formalizes $\Delta w_{i,j}$ as a product of the activations of pre- and post-synaptic activations, i.e.:

$$\Delta w_{i,j} = \eta \cdot a_i \cdot a_j \tag{4}$$

The plain Hebbian rule strengthens a synaptic efficiency when the signs of the pre- and post-synaptic neuron activations are positively correlated, weakens the synaptic efficiency when the signs are negatively correlated, and keeps the same efficiency when one of the two activations is zero (Brown et al., 1990). A constant η is used as a learning rate to scale the magnitude of the synaptic change.

As we stated in the previous section, one of the undesired effects of the plain Hebbian rule is that it may lead to an indefinite synaptic increase/decrease, since a synaptic change in one direction encourages further synaptic change in the same direction. A number of variants of the plain Hebbian rule have been proposed to stabilize the behavior (Vasilkoski et al., 2011; Brown et al., 1990). These rules can roughly be categorized into two groups: activity and threshold based. The activity based rules perform updates based on the activation correlations between pre- and post-synaptic neurons. The threshold-based methods perform updates when the activation correlations are above/below a given threshold, that can also be adaptive.

Finally, it is worth mentioning a further generalization of the plain Hebbian rule given by:

$$\Delta w_{i,j} = \eta \cdot [A \cdot a_i \cdot a_j + B \cdot a_j + C \cdot a_i + D] \quad (5)$$

This rule, usually referred to as the *ABCD* rule (Niv et al., 2002; Soltoggio et al., 2008), parameterizes the relationship between a_j and a_i by using four separate coefficients A, B, C, D.

2.2 Evolution of learning

The back-propagation algorithm is one of the conventional methods to train ANNs; however, it is considered as a biologically implausible method since it requires propagating back the error through synapses (Kuriscak et al., 2015). The biological evidence supports instead forms of Hebbian learning (Bienenstock et al., 1982; Brown et al., 1990; Sejnowski and Tesauro, 1989). Thus, Hebbian learning is often used as a learning mechanism for self-adapting ANNs based on supervision, and/or reinforcement signals, or in an unsupervised fashion (Soltoggio et al., 2018; Floreano and Urzelai, 2000; Hoerzer et al., 2014; Niv et al., 2002).

A near-optimum Hebbian rule and optimal learning parameters can depend on the task and the data. Furthermore, the parameters of the learning algorithm can affect the stabilization of the Hebbian rule and, ultimately, the performance of the ANN (Vasilkoski et al., 2011). Therefore, some previous works have proposed the use evolutionary computing to optimize the parameters of Hebbian learning rules (Coleman and Blair, 2012). Floreano and Urzelai (2000) evolved Hebbian learning rules for neural controllers in unsupervised settings by randomly initializing the parameters of networks, and letting the learning algorithm perform synaptic changes with a certain frequency during the network's lifetime. Niv et al. (2002) optimized the parameters of the Hebbian learning rules for an artificial bee foraging task in reinforcement learning settings.

Some of the other existing works considered replacing the Hebbian rules with evolving complex functions (i.e. ANNs) to perform synaptic changes. For instance, Orchard and Wang (2016) compared plastic networks where synaptic updates were performed using the ABCD rule against non-plastic networks; however, in their work the initial weights of the networks were also evolved by representing them into the genotype of the individuals. More recently, Risi and Stanley (2010) evolved a special kind of network (known as the *compositional pattern producing network, CPPN* (Stanley, 2007)) to perform synaptic changes.

Others used neuromodulated learning where a number of special neurons within the network are used for signaling synaptic updates to the other neurons in the network (Runarsson and Jonsson, 2000). Soltoggio et al. (2008) evolved network topologies and connection weights involving neuromodulated neurons and using the ABCD learning rule to perform synaptic updates. Tonelli and Mouret (2013) investigated the relation between different kinds of genetic encoding and neuromodulated learning capabilities of the networks.

In some problem settings (i.e. maze), the reinforcement signals may not be available after each action of the networks; rather, they may arrive after a sequence of actions of the networks over certain period of time. In this case, it may not be possible to associate the activations of neurons with the reinforcement signals. Yaman et al. (2019) proposed evolving synaptic plasticity rules that take into account stored activations of the neurons in each synapse to perform synaptic changes after certain period of time.

3 Evolution of Plasticity Rules

To evolve plasticity rules, we employ an evolutionary approach similar to the one proposed in (Yaman et al., 2019), which focuses on a case where the reinforcement signals are not immediately available after each action of the network. However, in this work, we focus on plasticity in a continuous learning scenario where the synaptic updates are performed based on the activations of neurons and reinforcement signals received immediately after each action of the networks. We represent the plasticity rules within the genotype of the individuals, and use Genetic Algorithms for the evolutionary process. The plasticity rules are evaluated based on their success in demonstrating robust learning capabilities on a foraging task with reinforcement learning settings under changing environmental conditions.

We use Equation (1) to compute the activation of each post-synaptic neuron, and an activation function given by:

$$\psi(x) = \begin{cases} 1, & \text{if } x > 0; \\ 0, & \text{Otherwise.} \end{cases} \tag{6}$$

which reduces the possible activation states of each neuron into two possibilities: *active* (1) and *passive* (0). As discussed below, this binary neuron model facilitates the interpretability of the results.

As for the modulatory signal, we use reinforcement signals received directly from the environment, in the following form:

$$m = \begin{cases} +1, & \text{if desired output (reward);} \\ -1, & \text{if undesired output (punishment);} \\ 0, & \text{otherwise (neutral).} \end{cases} \tag{7}$$

According to this signal, if the network produces the "desired" outcome then m is set to $+1$; if the network produces an "undesired" outcome then m is set to -1; otherwise m is set to 0. The desired and undesired outcomes are defined by a *reward function* which depends on the task, according to the desired/undesired associations between sensory inputs and behavioral outputs. The reward functions we used in our experiments are discussed in Section 4.1.

The initial values of the synaptic efficiencies are randomly sampled from the range of $[-1, 1]$ with uniform probability. After each network computation at time t, the synaptic efficiencies between post- and pre-synaptic neurons at time $t + 1$ are updated based on the following rule:

$$w'_{i,j}(t+1) = w_{i,j}(t) + \eta \cdot f(a_i, a_j, m) \tag{8}$$

where $f()$ is a "composed" Hebbian plasticity rule that determines how each synaptic efficiency $w_{i,j}$ is updated depending on the activations of the pre- and post-synaptic neurons, a_j and a_i, and the modulatory signal m. Differently from the existing approaches described in Section 2.2, here we evolve *discrete rules* to specify the synaptic modifications corresponding to each possible combination of the pre- and post-synaptic activations and the modulatory signal, as shown in the table illustrated in Figure 1. It is important to note that it is possible to enumerate all the possible discrete rules as we chose a binary neuron model, see Equation (6), and a discrete modulatory signal, see Equation (7).

Finally, after each update provided by Equation (8), the weights are scaled as follows:

$$w'_{i,j} = \frac{w'_{i,j}}{||\boldsymbol{w'_i}||_2} \tag{9}$$

where the row vector $\boldsymbol{w'_i}$ encodes all incoming weights to a post-synaptic neuron i. This scaling allows to have a unit length using the Euclidean norm $||\cdot||_2$. Furthermore, this normalization process prevents indefinite synaptic growth, and helps connections' specialization by introducing local synaptic competition, a concept recently observed also in biological neural networks (El-Boustani et al., 2018).

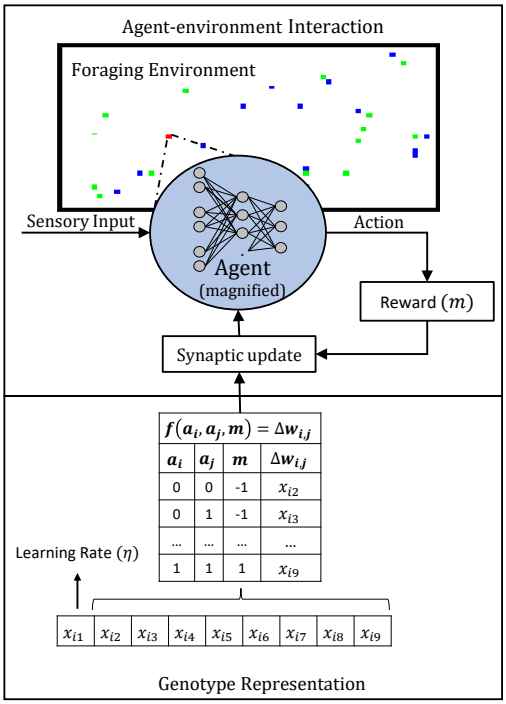


Figure 1: Genotype representation and agent-environment interaction. The genotypes of the individuals encode the learning rate and synaptic update outcomes for 8 possible states of a_i, a_j and m. The agent (color coded as red) is evaluated on a foraging task where it is expected to learn to collect/avoid correct types of items (color coded as blue and green). An artificial neural network is used to perform the actions of the agent. The initial weights of the ANN are randomly initialized, and are updated after each action step based on the evolved plasticity rules and a reinforcement signal received from the environment.

3.1 Details of the Genetic Algorithm

We employ a *Genetic Algorithm (GA)* to optimize the plasticity rules (Goldberg, 1989). The genotype of the individuals, illustrated in Figure 1, encodes the learning rate $\eta \in [0, 1)$, and one of three possible outcomes $L = \{-1, 0, 1\}$ (corresponding to *decrease*, *stable*, and *increase*, respectively) for each of the plasticity rules defined by $f(a_i, a_j, m)$.

Since we use binary activations for neurons, there are 4 possible activation combinations for a_i and a_j. Furthermore, we take into account only positive and negative modulatory signal states ignoring $m = 0$ because the synaptic update is performed only when $m = +1$ or $m = -1$. Consequently, there are 8 possible input combinations, and therefore 8 possible plasticity rules defined by $f(a_i, a_j, m)$. The size of the search space of the plasticity rules is then 3^8, excluding the real-valued learning rate parameter.

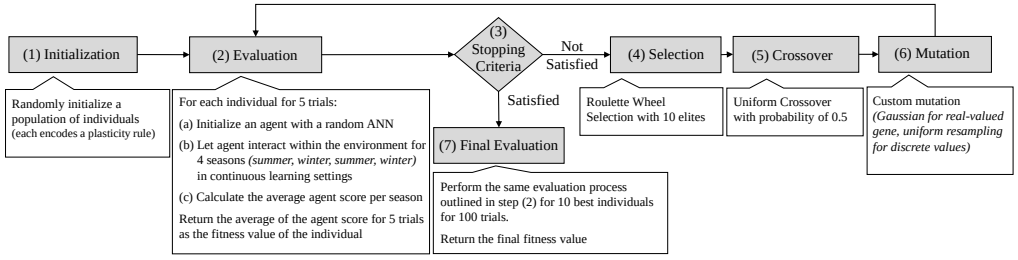


Figure 2: Graphical illustration of the steps of the Genetic Algorithm used to evolve the plasticity rules.

A graphical illustration of the GA is provided in Figure 2. In the initialization step of the algorithm, we randomly initialize a population of 9-dimensional individual vectors $\boldsymbol{x}_i$, where $i = (1, \ldots, N)$ (in our experiments, N was set to 30) to encode synaptic update rules. Each dimension of the individuals are uniformly sampled within their domain depending on their data type.

The evaluation process of an individual starts with the initialization of an agent with a random ANN configuration. Here, we use fixed topology fully connected feed forward neural networks, and sample the connection weights from a uniform distribution in [-1,1]. The agent is allowed to interact with the environment by performing actions based on the output of its controlling ANN. After each action step, the weights of the ANN are updated based on the synaptic update table. This table is constructed by converting the vector representation of the individual plasticity rules to specify how synaptic weights are modified based on the pre-, post-synaptic, and modulatory signals (see Figure 1). After every agent's action, a reinforcement signal is received from the environment and used directly as the modulatory signal.

We define a certain number of actions to allow the agent to interact with the environment. This process is divided into four periods of equal lengths, which we refer to as "seasons". We calculate the average of the agent's performance score, per each season, by subtracting the number of incorrectly collected items from the number of correctly collected items. Due to the stochasticity of this process (because of the random network initialization), we perform this process, starting from the same initial conditions, for five independent trials. Thus, the fitness value of an individual plasticity rule is given by:

$$fitness = \frac{1}{s \cdot t} \sum_{k=1}^{s} \sum_{l=1}^{t} (c_{k,l} - i_{k,l}) \tag{10}$$

where t and s are the number of trials and seasons, respectively, and $c_{k,l}$ and $i_{k,l}$ are the number of correctly and incorrectly collected items in each season k of each trial l.

The evaluation process of the GA is based on a continuous learning setting, where the weights of the ANNs are updated constantly. Thus, the algorithm does not involve

any validation to store the best network configurations. However, in our experiments we additionally tested the evolved plasticity rules with validation settings, in order to show that validation improves the performance of the agents. On the other hand, this additional step is computationally expensive since it requires the agents to be tested for a certain number of further action steps.

We use an *elitist roulette wheel selection* operator to determine parents of the next population. The top 10 *elites* (best individuals) are copied to the next generation without any change. The rest of the offspring are generated using a *uniform crossover* operator with a probability of 0.5. As for the mutation operator, we perturb the real-valued component by a small value sampled from a Gaussian distribution with 0 mean and 0.1 standard deviation, and re-sample the discrete components with a probability of 0.15.

The evolutionary process is executed until there is no more improvement in terms of best fitness for a certain number of evaluations. At the end of the evolutionary process, the top 10 elite individuals in the population are evaluated for 100 trials, and the average statistics of this process is provided as the final result of the Genetic Algorithm. We execute the GA for 30 independent runs.

4 Experimental Study

We test the learning and adaptation capabilities of the plastic ANNs with evolved plasticity rules on an agent-based foraging task within a reinforcement learning setting inspired by Soltoggio and Stanley (2012). In this task, an artificial agent, operated by a plastic ANN, is required to learn to navigate within an enclosed environment and collect/avoid correct types of items placed in the environment, based only on the reinforcement signals received in response to its actions.

The foraging environment contains two types of food items and is enclosed by a wall. To test the adaptation abilities of the networks, we define two reward functions that we refer to as two seasons: *summer* and *winter*. In both seasons, the agent is expected to explore the environment and avoid collisions with the walls. During the summer season, the agent is expected to learn to collect and avoid specific types of objects, while the expectation for the types of objects is swapped during the winter season. The details of the foraging task and environment are provided in the following section.

4.1 Foraging task and environment

A visualization of the simulated environment used in the foraging task is provided in Figure 3. We initialize a 100x100 grid enclosed by a wall. In the following, we refer to the green and blue items, and the wall, as "G", "B" and "W" respectively. In the initialization phase of an experiment, an agent, 50 "G" and 50 "B" items are randomly placed on the grid.

The architecture of the ANNs used to control the agent is shown in Figure 4. The agent is located in a cell, and has a direction to indicate its orientation on the grid. It is equipped with three sensors that can take inputs from the nearest cell on the left, in front, and on the right. Since there are four possible states for each cell (nothing, "W", "G", "B"), we represent the sensor reading of each cell with two bits, as $[(0,0),(1,1),(1,0),(0,1)]$. The agent performs one of the three possible actions as "Left", "Straight", and "Right" based on the output layer of its ANN. The output neuron with the maximum activation value is selected to be the action of the agent. In cases of "Left" and "Right" the agent's direction is changed accordingly, and the agent is moved one cell in the corresponding direction. In case of "Straight", the direction of

the agent is kept constant and the agent is moved one cell along its original direction.

We use fully connected feed forward networks with one hidden layer with 6 input, 20 hidden and 3 output neurons. We also perform additional experiments for networks consisting of various numbers of hidden neurons (see next section for details). Additional bias neurons are added to the input and hidden layers.

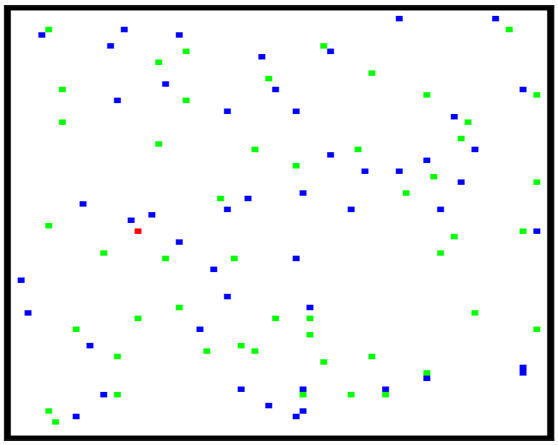


Figure 3: Simulation environment used in the agent-based foraging task experiments. The location of the agent, two types of items and the wall are color coded with red, green, blue and black respectively.

The statistics of the agent while interacting with the environment are collected. If the agent visits a cell occupied by a "G" or "B" item, the item is collected and recorded. A collected item disappears from its original location and reappears in another location selected at random. If the agent steps into a cell occupied by a wall, the action is ignored and recorded as a hit on the wall. After each action, a reinforcement signal is received from the environment, and used as a modulatory signal, see Equation (7).

The complete list of sensory states, behaviors and reinforcement signal associations that we used in our experiments is provided in Table 1. In the table, the columns labeled as "Sensor" and "Behavior" show the sensor states and actions of the network, respectively. The reward functions corresponding to the two seasons, "Summer" and "Winter", are also shown. It should be noted that for some sensory states, multiple reward function associations may be triggered. In these cases, the associations that concern the behaviors to collect/avoid items are given priority. We should also note that the reward functions described here do not specify the reinforcement signal outcomes for *all* possible sensor-behavior combinations. Indeed, in total there are 192 possible sensor-behavior associations (resulting from 3 possible behavior outcomes for each of 2^6 possible sensor states).

The first two reward associations are defined to encourage the agent to explore the environment. Otherwise, the agent may get stuck in a small area, for example by performing only actions such as going left or right when there is nothing present. Sensor state labeled as "nothing" refers to an input to the network equal to $[0, 0, 0, 0, 0, 0]$.

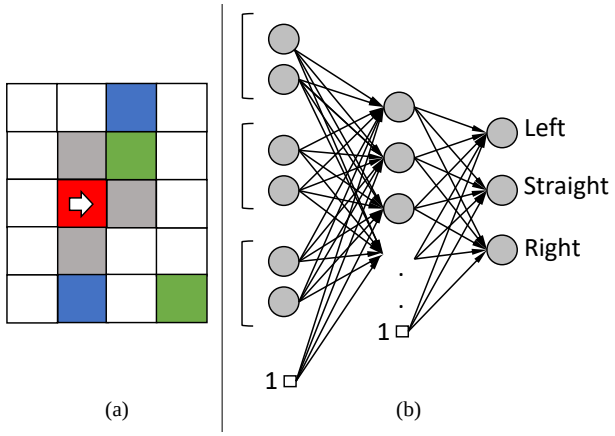


Figure 4: Details of the agent-based foraging task experiments: (a) Agents' position/direction (red cell) and sensory inputs from left, front and right cells (in gray) within a foraging environment containing two types of items (green and blue) items; (b) The ANN controller, with a hidden layer, 6 inputs (2 per cell), and 3 outputs (left/right/straight).

Reward associations 3 through 8 specify the reinforcement signals for the behaviors in relation to the wall, and are the same in both summer and winter seasons. It is expected that the agent learns to avoid the wall. Therefore, we define positive reward signals for the states where there is a wall, and the agent picks a behavior that avoids a collision (IDs: 3, 5, 6). Conversely, we define punishment signals for the sensor-behavior associations where the agent collides to the wall (IDs: 4, 6, 8). For instance, the sensor state of the association given in ID. 3 refers to the input to the network as $[0, 0, 1, 1, 0, 0]$, and provides reward if the agent decides to go left or right (and, this behavior is desired in both summer and winter seasons).

Reward associations 9 through 14 and 15 through 20 define sensor-behavior and reinforcement signal associations regarding the "G" and "B" types of items respectively. The reinforcement signals of the seasons are reversed for these two seasons. In summer, the agent is expected to collect "G" items and avoid "B" items, whereas in winter the agent is expected to collect "B" items and avoid "G" items.

We perform an experiment by first placing into the environment an agent, initialized with a random ANN, and starting with the summer season. We then allow the agent to interact in a continuous learning setting for 5000 action steps. At the end of the summer season, we let the agent continue its interaction with the environment for another 5000 action steps by keeping its network configuration and only changing the season to winter. At the end of the given number of action steps, we perform two more seasonal changes in the same fashion. The performance of the agent is the average of the performances over the four seasons.

For all experiments, the agents are set to pick a random action with a probability of 0.02 regardless of the actual output of their ANNs. Random behaviors are introduced to avoid getting stuck in a behavioral cycle. Such random behaviors, though, are not taken into account in the synaptic update procedure.

Table 1: Associations of the Sensor and Behavior states to the reinforcement signals for Summer and Winter seasons.

ID.	Sensor	Behavior	Summer	Winter
1	nothing	straight	1	1
2	nothing	left or right	-1	-1
3	W straight	left or right	1	1
4	W straight	straight	-1	-1
5	W on left	right	1	1
6	W on left	left or straight	-1	-1
7	W on right	left	1	1
8	W on right	right or straight	-1	-1
9	G straight	straight	1	-1
10	G straight	left or right	-1	0
11	G on left	left	1	-1
12	G on left	straight or right	-1	0
13	G on right	right	1	-1
14	G on right	straight or left	-1	0
15	B straight	straight	-1	1
16	B straight	left or right	0	-1
17	B on left	left	-1	1
18	B on left	straight or right	0	-1
19	B on right	right	-1	1
20	B on right	straight or left	0	-1

5 Experimental Results

In this section, we discuss the results of the evolved plasticity rules evolved in our experiments. We then show the learning and adaptation processes of the best performing evolved plasticity rule during a foraging task in continuous learning settings, with and without validation. Finally, we show additional experimental results reporting the best performing evolved plasticity rules used with ANNs with various number of hidden neurons.

For comparison, we provide the results of two algorithms: the results of agents with ANNs controllers that are optimized by using the Hill Climbing (HC) algorithm (De Castro, 2006), and the results of a rule-based agent that is controlled by hand-coded rules without using an ANN.

In the case of the HC algorithm, we start with an ANN (the same architecture as used in the plasticity experiments) where all of its weights are randomly initialized between $[-1, 1]$, evaluated on our foraging task (using the same evaluation settings for 5000 action steps) for the first summer season to find its fitness, and assigned as the best network. We then iteratively generate a candidate network by randomly perturbing all the weights of the best network using Gaussian mutation with zero mean 0.1 standard deviation, evaluate on our foraging task for a given season, and replace the best network with the candidate network if its fitness value is better than the fitness of the best network. We repeat this process consecutively for summer, winter, summer and winter seasons for 1000 iterations each, in total of 4000 iterations combined. When a seasonal change happens, we keep the best network and continue the optimization procedure as specified.

Figure 5 shows the average results of 100 runs (100×4000 iterations) of the HC algorithm for four seasons. The average values and standard deviations of fitness values,

collected number of Green and Blue items, and Wall hits of the 10th, 500th and 1000th iterations for each season are given in Table 2. The values for Summer1, Winter1, Summer2 and Winter2 presented in table corresponds to the iteration numbers in ranges 0-1000, 1001-2000, 2001-3000 and 3001-4000 on x-axis of Figure 5 respectively.

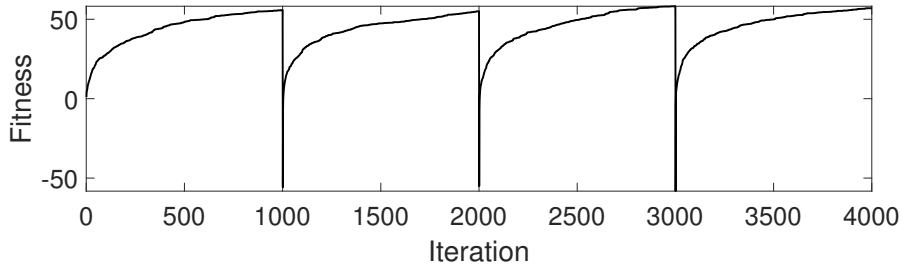


Figure 5: The average fitness value during 100 runs of the offline optimization process of the agents using the HC algorithm without the plasticity property. The process starts with a randomly initialized ANN which is trained on our foraging task starting with the summer season, and every 1000th iteration, the seasons is changed.

Table 2: The average and standard deviations of the performance statistics of 100 runs of the offline optimization procedure of the agents using the Hill Climbing algorithm.

Result	Iteration	Summer1	Winter1	Summer2	Winter2
Fitness	**10th**	9.20 ± 10.12	9.79 ± 11.83	8.74 ± 10.24	10.41 ± 12.04
	500th	48.34 ± 24.82	47.42 ± 19.15	49.59 ± 18.02	49.94 ± 17.04
	1000th	55.81 ± 23.04	55.08 ± 20.00	58.21 ± 17.08	56.95 ± 18.35
G	**10th**	11.61 ± 12.37	5.02 ± 7.63	13.32 ± 12.22	4.48 ± 6.46
	500th	49.36 ± 24.48	1.58 ± 4.87	51.71 ± 17.43	1.59 ± 4.62
	1000th	56.56 ± 22.88	0.85 ± 3.29	59.65 ± 16.89	1.38 ± 4.09
B	**10th**	2.41 ± 5.59	14.81 ± 14.40	4.58 ± 6.66	14.89 ± 14.05
	500th	1.02 ± 3.23	49.00 ± 19.06	2.12 ± 4.73	51.53 ± 16.33
	1000th	0.75 ± 1.43	55.93 ± 19.88	1.44 ± 3.47	58.33 ± 17.48
W	**10th**	614.09 ± 919.95	693.78 ± 854.64	770.47 ± 1101.77	702.46 ± 862.67
	500th	170.26 ± 463.02	157.15 ± 446.59	146.47 ± 417.37	183.05 ± 485.90
	1000th	114.92 ± 375.96	88.30 ± 351.09	115.72 ± 398.26	140.97 ± 439.384

We observe that it takes about 1000 iteration for the networks to reach 56.51 fitness value on average at the end of each season. Since the networks are well optimized for the task, a sudden decrease in their performance at the beginning of each seasonal change is observed. In about 10 generations, the fitness values are increased around 9 in each season. We see clear increasing and decreasing trends in number of collected items depending on the season while the number of iterations increase. Moreover, the number of wall hits are also reduced although it is relatively high even at the end of the iterations for each season.

This agent that is controlled by hand-coded rules without ANNs has a "perfect knowledge" of which items to collect/avoid in each season throughout its evaluation process. Also, it moves straight if there is nothing around, and makes a turn when it encounters a wall (this behavior is expected to improve the exploration of the environ-

ment). The mean and standard deviation of the rule-based agent for 100 trials are given in Table 3. Since, there is no learning or optimization process involved in this case, we only show its results at the end of a Summer and Winter seasons.

Table 3: The statistics of the hand-coded rule-based agent with perfect knowledge of the task in each season.

	Summer				**Winter**			
	Fitness	**G**	**B**	**W**	**Fitness**	**G**	**B**	**W**
Mean	67.2	67.2	0	0	67.8	0	67.8	0
Std. D.	8.6	8.6	0	0	7.9	0	7.9	0

5.1 Evolved versus defined plasticity rules

We collected a total of 300 evolved plasticity rules by performing 30 independent GA runs. The max, min, median, mean and standard deviation of the average fitness values of all evolved plasticity rules are 49.96, 7.97, 47.37, 37.89 and 13.98 respectively.

Table 4: Statistics of the complete list of distinct evolved plasticity rules found by the GA, ranked by their median fitness. The columns "Number of Evolved Rules" shows the number of evolved rules found for each distinct rule; "Median", "Std.D", "Max", "Min" show their median, standard deviation, max and minimum fitness; "η Mean", "η Std.D." show their average learning rate and standard deviations, respectively. The rest of the columns, encoded in 2-bits, represent the activation states of pre- and post-synaptic neurons when modulatory signal is -1 or 1.

								$m=-1$				$m=1$			
ID.	**Number of Evolved Rules**	**Median**	**Std.D.**	**Max**	**Min**	**η Mean**	**η Std.D.**	**00**	**01**	**10**	**11**	**00**	**01**	**10**	**11**
1	164	48.63	0.83	49.96	42.95	0.0375	0.008	0	0	1	-1	0	0	0	0
2	23	44.23	1.05	45.88	41.55	0.0167	0.004	-1	1	1	-1	0	0	0	0
3	3	42.35	2.65	46.45	41.48	0.0192	0.003	1	0	1	-1	0	0	0	0
4	19	28.35	0.51	29.05	27.22	0.0488	0.009	-1	1	1	-1	1	0	-1	0
5	1	27.28	0	27.28	27.28	0.0182	0	-1	-1	1	0	1	0	-1	0
6	9	26.70	1.48	27.91	22.80	0.0118	0.003	-1	1	1	-1	-1	-1	1	1
7	16	26.54	0.89	28.13	24.65	0.0092	0.002	0	1	1	-1	-1	-1	1	1
8	2	25.91	0.07	25.97	25.86	0.0096	0.0008	1	1	1	-1	-1	-1	1	1
9	1	23.55	0	23.55	23.55	0.0273	0	-1	1	1	-1	0	-1	0	1
10	2	22.11	1.26	23	21.21	0.0052	0.003	1	1	1	-1	0	-1	0	1
11	10	20.96	0.33	21.59	20.45	0.0198	0.003	0	1	1	-1	1	-1	-1	-1
12	20	20.63	0.40	21.34	19.81	0.061	0.022	-1	1	1	0	1	-1	-1	-1
13	1	12.41	0	12.41	12.41	0.0799	0	0	0	0	-1	1	1	0	-1
14	19	10.30	0.55	10.87	8.36	0.0662	0.018	0	0	0	-1	0	1	1	-1
15	10	8.82	0.54	9.59	7.97	0.0301	0.009	1	1	-1	-1	1	0	-1	1

Table 5: The statistics (fitness values, standard deviations of the fitness values, number of correctly and incorrectly collected items and their standard deviations, and number of wall hits and its standard deviation) of some rules defined by hand over 100 trials. Columns encoded in 2-bits represent the activation states of pre- and post-synaptic neurons.

										$m=-1$				$m=1$			
ID.	**Fitness**	**Std Fit.**	**Correct**	**Std Cor.**	**Incor.**	**Std Inc.**	**Wall**	**Std Wall**	η	**00**	**01**	**10**	**11**	**00**	**01**	**10**	**11**
16	41.68	18.26	47.63	14.36	5.95	7.17	14.36	54.59	0.04	0	0	0	-1	0	0	0	0
17	5.6	9.28	25.2	6.88	19.7	5.28	120	140.2	0.01	0	0	1	-1	0	-1	0	1
18	0.2	6.13	18.3	4.48	18.1	4.27	602.6	117.5	0.01	0	0	0	-1	0	0	0	1

Table 6: Statistics of the best evolved rule on continuous learning for each season.

	Summer 1				**Winter 1**				**Summer 2**				**Winter 2**			
	Fitness	**G**	**B**	**W**	**Fitness**	**G**	**B**	**W**	**Fitness**	**G**	**B**	**W**	**Fitness**	**G**	**B**	**W**
Mean	50.81	53.37	2.56	5.53	48	4.13	52.1	3.6	50.37	53.71	3.3	3.18	50.61	3.7	54.33	4.7
Std. Dev.	10	9.5	1.8	3.5	9.2	2.2	8.4	3.6	10.2	9.64	1.7	3.4	10.2	1.9	9.5	6.3

We identified in total of 15 distinct rules, distinguished only by their discrete part (i.e., without considering the specific value of the learning rate). A complete list of these rules is provided Table 4. The first and second column show the distinct rule identifier (ID) and the number of rules found for each distinct rule, respectively. The columns labeled as "Median", "Std.D.", "Max", "Min", "η Mean", and "η Std.D." show the median, standard deviation, max and min values of the fitness, and the average and standard deviations of the learning rates per each distinct rule respectively. In the remaining columns, we report the activation states of pre- and post-synaptic neurons a_j and a_i (encoded as 2-bits) when $m = -1$ and $m = 1$. The most successful rules, shown in the first three rows, constitute 63% of all collected rules, and achieve a fitness values greater than 40. They perform synaptic updates only when $m = -1$. The rest of the rules perform synaptic updates also when $m = 1$. However, their performance is significantly worse in terms of their fitness values. For instance, the rules given in the fourth row achieve the best fitness value of 29.05 which is 20 points lower than the best rule.

The overall best performing evolved plasticity rule (the best of 164 rules with ID. 1 shown in Table 4) achieved a fitness value of 49.96 with a standard deviation of 9.97, corresponding to an average of 53.39 correctly collected and 3.43 incorrectly collected items, with standard deviations of 9.31 and 2.04 items respectively, and on average incurred 4.27 wall hits, with a standard deviation of 4.46 hits. The learning rate of this rule is $\eta = 0.039$. This rule performs synaptic updates in two cases only, i.e. when the network produces undesired behavior $(m = -1)$. In the first case, the plasticity rule increases the synaptic weights between a_j and a_i when a_j is active but a_i is not. This may facilitate finding new connections. The second case implements an anti-Hebbian learning where the plasticity rule decreases the synaptic weights when both a_j and a_i are active.

For comparison, Table 5 provides the statistics of some plasticity rules that we defined by hand. The columns "Fitness", "Std Fit.", "Correct", "Incor.", "Std Cor." and "Std Inc." show the fitness, standard deviation of the fitness, average number of correctly and incorrectly collected items, and their standard deviations, respectively. The remaining columns show the details of the plasticity rule.

The rule with ID. 16 was defined by taking the best performing evolved rule (ID. 1) and replacing the part corresponding to $m = -1$, activations=10 with 0. After this change, the rule performs synaptic updates only when pre- and post-synaptic neurons are active and the network produces an undesired outcome ($m = -1$). When this rule is used, the performance of the networks are significantly better than other rules defined by hand. However, the performance is worse than the rule without this replacement. Surprisingly though, the GA did not find this rule, as shown in the distinct rule list given in Table 4, even though it performs better than most of all the other rules. This may be due to the convergence of the evolutionary process to the rules that perform better, specifically the rules given in the first three rows.

The rules given in row IDs. 17 and 18 were also defined by hand, but showed the worst performances. In particular, the rule given in row ID. 18 performs Hebbian/anti-Hebbian learning as it increases/decreases the synaptic weights between neurons when they are both active and the network produces a desired/undesired outcome. Instead, the rule given in row ID. 17 performs synaptic updates on two additional activation combinations w.r.t. rule with ID. 18, in order to facilitate the creation of new connections. Even though this latter rule performs better than the rule ID. 18, its performance is not better than the evolved plasticity rule with the worst performance.

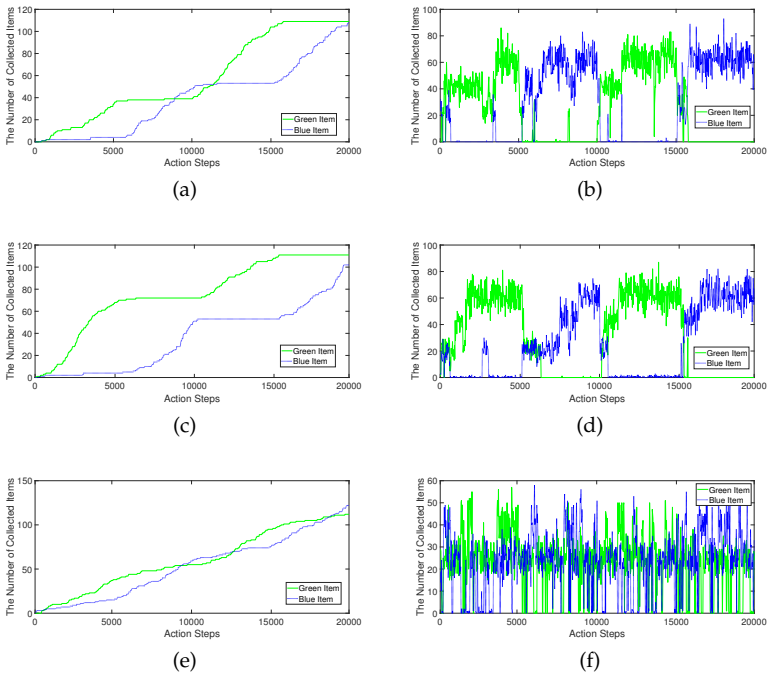


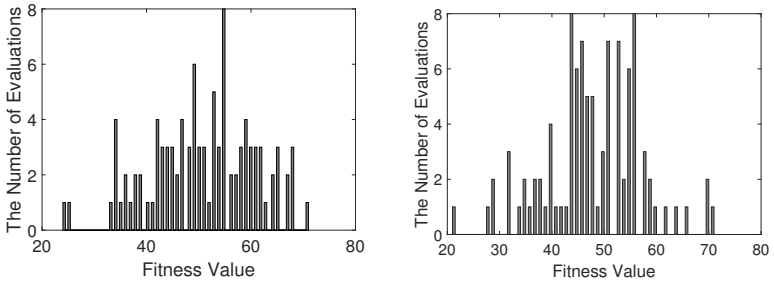
Figure 6: The statistics of three selected plasticity rules during a single run of a foraging task over four seasons. Each row of the figures provides the results of the best plasticity rules from the rule types with IDs. 1, 2, and 6 respectively. Figures in the first column show cumulative results of the number of items collected, whereas figures in the second column show the number of items collected when the agent is tested independently for 5000 action steps using the configurations of its ANN at the time of the measurement.

The reason why the performance of the plasticity rules that perform synaptic updates also when $m = 1$ is worse than the ones that perform synaptic update only when $m = -1$ is likely due to the design of the task and the reward function. Since the reward function keeps providing rewards while a network is achieving the desired outcomes, the accumulation of these rewards can "overcharge" the synaptic weights and may cause forgetting after a certain number of synaptic updates. The issue of forgetting already known knowledge/skills due to the acquisition of new knowledge/skills in ANNs is usually referred as "catastrophic forgetting" (Parisi et al., 2019).

5.2 Performance of the networks during a foraging task

Figure 6 shows the statistics of some selected plasticity rules during a single run of a foraging task over four seasons. The overall process lasts 20000 action steps in total, consisting of four seasons of 5000 action steps each. The foraging task starts with the summer season and switches to the other season every 5000 action steps. Measurements were sub-sampled at every 20th action step to allow better visualization. The figures given in the first column show the cumulative number of items of both types collected throughout the process, whereas each measurement given in the figures in the second column shows the number of items collected when the agent is tested independently for 5000 action steps using its ANN configuration at the time of the mea-

surement.



(a) Summer 1 (b) Winter 1

(c) Summer 2 (d) Winter 2

Figure 7: The distribution of fitness values of the best evolved rule from ID.1 in Table 4 over four seasons. The x- and y-axes of each subfigure show the fitness value and number of evaluations, respectively.

Figures 6a and 6b show the results of the best rule from ID. 1, Figures 6c and 6d show the results of the best rule from ID. 2, and Figures 6e and 6f show the results of the best rule from ID. 6. In the case of the rules from IDs. 1 and 2, the agent learns quickly to collect the correct type of items in each season. The number of correctly collected items from the previous season stabilizes, and the number of incorrectly collected items from the previous season increases in the next season after a seasonal change occurs. We observe in Figures 6b and 6d that there are distinct and stable seasonal trends for each season. The noise in the measurements is due to the sub-sampling and stochasticity of the evaluation process.

In the case of rule from ID. 6, the agent keeps collecting both kinds of items; however, the number of correctly collected type of items is larger than incorrectly collected items in each season. We can observe in Figure 6f that the testing performance is not stable throughout the process. The fluctuations show that the configuration of the ANN seems to be frequently shifting the states between learning and forgetting.

Table 6 and Figure 7 show, respectively, the average statistics and the distribution of the fitness of the best performing evolved plasticity rule in each season over 100 trials. The agent achieves an average fitness value of about 50 in all the seasons except the first winter season where it achieves a fitness value of 48.

Table 7 shows the average results of the best ANN configuration found through validation. During a foraging task process, the agent is tested independently at every 20th action step, and the configuration of the ANN that achieved the highest fitness

Table 7: The statistics of the best performing evolved rule with validation.

	Summer				Winter			
	Fitness	**G**	**B**	**W**	**Fitness**	**G**	**B**	**W**
Mean	61.0	61.0	0	0	63.0	0	63.0	0
Std. D.	8.62	8.62	0	0	8.19	0	8.19	0

value was stored as the best ANN configuration.

To assess the statistical significance of the difference of the results produced by different agents, we use the Wilcoxon rank-sum test (Wilcoxon, 1945). The null-hypothesis, i.e. that the mean of the results produced by two agents (thus, their behavior) are the same, is rejected if the p-value is smaller than $\alpha = 0.05$. We perform pairwise comparisons of three sets of results of fitness values for summer and winter seasons obtained from three agents evaluated over 100 evaluations. More specifically, we compare the results of the hand-coded rule-based agent, with those of the agents that use the best performing evolved plasticity rule in continuous learning settings, with and without validation. Based on the pairwise comparison results, the hand-coded rule-based agent is significantly better than the other two agents with and without validation, with significance levels of $1.9 \cdot 10^{-28}$ and $8.2 \cdot 10^{-05}$ respectively. Furthermore, the results of the agent with validation are significantly better than those without validation, with a significance level of $9.9 \cdot 10^{-14}$.

To gain further insight into these results, we have recorded the behavior of the agent during a foraging tasks with four seasons each consisting of 3000 action steps[1]. The demonstration shows that the agent is capable of efficiently adapting to the environmental conditions imposed by each season. Even though the desired behavior with respect to the wall should be constant across the different seasons, the agent makes a few mistakes at the beginning of each season by hitting the wall. This may be due to the change of the synaptic weights that affect the behavior of the agent with respect to the wall. Furthermore, we provide the actual weights and their change after each seasonal change in A.

5.3 Sensitivity analysis on the number of hidden neurons

Finally, we performed a sensitivity analysis of our results with respect to the number of hidden neurons. In Figure 8, we show the average results of 100 trials each performed using the best performing evolved plasticity rule on the ANNs, with various number of hidden neurons. We test networks with 5, 10, 15, 20, 25, 30, 35, 40, 45 and 50 hidden neurons. The results show that there is a 13 point increase on the average fitness value when the number of hidden neurons is increased from 5 to 20. There is also a slight upward trend when larger than 20 hidden neurons are used, which results in a 3 point average fitness gain when the number of hidden neurons is increased from 20 to 50. Moreover, the standard deviations are also slightly reduced while the number of hidden neurons increases, showing that the use of more hidden neurons tends to make the agent's behavior more consistent across different trials.

[1] An online video demonstration of the behavior of an agent for four seasons each consisting of 3000 action steps: `https://youtu.be/9jy6yTFKgT4`.

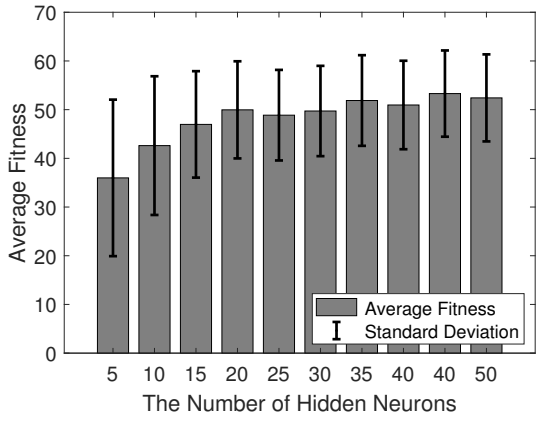


Figure 8: The average fitness values of the ANNs with various number of hidden neurons.

6 Conclusions

The plasticity property of biological and artificial neural networks enables learning by modifying the networks' configurations. These modifications take place at individual synapse/neuron level, based on the local interactions between neurons.

In this work, we have proposed an evolutionary approach to produce autonomous learning in ANNs in a task with changing environmental conditions. We devised plasticity rules to conduct synaptic adjustments inspired by the local plasticity property of the BNNs. The plasticity rules operate at individual synapse level, based on the interactions between artificial neurons. We employed Genetic Algorithms (GA) to explore the search space of the possible plasticity rules for all possible states of the post- and pre-synaptic neuron activations and reinforcement signals.

We evaluated the evolved plasticity rules on an agent-based foraging task, focusing specifically on the adaptation capabilities of the ANNs under changing environmental conditions. In our test scenario, an agent starts with a randomly initialized ANN configuration, and is required to learn collecting/avoiding correct types of items while interacting with the environment. After a certain number of action steps, the types of items to collect/avoid are switched, and the agent is expected to adapt to the new conditions.

The results of the multiple runs of the GA produced in total 300 rules, that we have grouped in 15 distinct plasticity rules (based only on the discrete parts of the rule), among all $3^8 = 6561$ possible rules. Interestingly, more than half of all plasticity rules (164 of 300) converged on a single type of rule that showed the best performance.

To set an upper and a lower bound for the performance of the ANNs with evolved plasticity rules, we performed a set of separate experiments with hand-coded rule-based agents with perfect knowledge and agents with ANN controllers that are optimized using the HC algorithm respectively. We also observe that (not reported in the paper) the fitness of an agent that takes decisions randomly at each action step is zero, as expected, since in this case the agent does not have any intelligent mechanism to distinguish between items to collect; thus, we did not report the performance results of this experimental configuration.

Comparison with a hand-coded rule-based agent with a perfect knowledge of the task showed instead that the performance of the best evolved rule produced agents that can perform the task very well (as good as about 74% of the performance of the hand-coded agent), considering a continuous learning setting where there are no separate resources for training and testing. In addition, we showed the efficiency of the evolved plasticity rules in searching the configuration space of the networks by performing validation during the learning process. With validation, the best networks could achieve 94% of the performance of the hand-coded rule-based agent (this performance difference may be due to the specific design of the reward function).

The results of the agents with ANNs that are trained using the HC algorithm shows that the average fitness value at 1000th iteration is better than the results of the agents with ANNs trained with the plasticity rules in continuous learning settings without validation. However, the results of the HC has a higher standard deviation. This is reasonable since in continuous learning settings, the learning process (thus the mistakes the agents make) are also involved into the evaluation process. On the other hand, the agents that are trained with plasticity rules with validation performed better than the agents that are trained using the HC.

Surprisingly, the best evolved rule performed synaptic updates only when the network produced undesired output (negative reinforcement signal). This may be due to the specific reward functions we used in the experimentation, which were designed to provide constant reward/punishment while the networks produced desired/undesired outcomes. Intuitively, after the networks learn to perform the task successfully, keep performing synaptic updates may cause degradation in the synaptic weights and result in forgetting.

7 Acknowledgements

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No: 665347.

References

Bienenstock, E. L., Cooper, L. N., and Munro, P. W. (1982). Theory for the development of neuron selectivity: orientation specificity and binocular interaction in visual cortex. *Journal of Neuroscience*, 2(1):32–48.

Brown, T. H., Kairiss, E. W., and Keenan, C. L. (1990). Hebbian synapses: biophysical mechanisms and algorithms. *Annual review of neuroscience*, 13(1):475–511.

Coleman, O. J. and Blair, A. D. (2012). Evolving plastic neural networks for online learning: review and future directions. In *Australasian Joint Conference on Artificial Intelligence*, pages 326–337. Springer.

De Castro, L. N. (2006). *Fundamentals of natural computing: basic concepts, algorithms, and applications*. CRC Press.

El-Boustani, S., Ip, J. P., Breton-Provencher, V., Knott, G. W., Okuno, H., Bito, H., and Sur, M. (2018). Locally coordinated synaptic plasticity of visual cortex neurons in vivo. *Science*, 360(6395):1349–1354.

Floreano, D., Dürr, P., and Mattiussi, C. (2008). Neuroevolution: from architectures to learning. *Evolutionary Intelligence*, 1(1):47–62.

Floreano, D. and Urzelai, J. (2000). Evolutionary robots with on-line self-organization and behavioral fitness. *Neural Networks*, 13(4-5):431–443.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition.

Hebb, D. O. (1949). The organization of behavior: A neuropsychological theory.

Hoerzer, G. M., Legenstein, R., and Maass, W. (2014). Emergence of Complex Computational Structures From Chaotic Neural Networks Through Reward-Modulated Hebbian Learning. *Cerebral Cortex*, 24(3):677–690.

Kowaliw, T., Bredeche, N., Chevallier, S., and Doursat, R. (2014). Artificial neurogenesis: An introduction and selective review. In *Growing Adaptive Machines*, pages 1–60. Springer.

Kuriscak, E., Marsalek, P., Stroffek, J., and Toth, P. G. (2015). Biological context of Hebb learning in artificial neural networks, a review. *Neurocomputing*, 152:27–35.

Mocanu, D. C., Mocanu, E., Stone, P., Nguyen, P. H., Gibescu, M., and Liotta, A. (2018). Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature Communications*, 9(1):2383.

Mouret, J.-B. and Tonelli, P. (2014). Artificial evolution of plastic neural networks: a few key concepts. In *Growing Adaptive Machines*, pages 251–261. Springer.

Niv, Y., Joel, D., Meilijson, I., and Ruppin, E. (2002). Evolution of Reinforcement Learning in Uncertain Environments: A Simple Explanation for Complex Foraging Behaviors. *Adaptive Behavior*, 10(1):5–24.

Nolfi, S., Miglino, O., and Parisi, D. (1994). Phenotypic plasticity in evolving neural networks. In *From Perception to Action Conference, 1994., Proceedings*, pages 146–157. IEEE.

Orchard, J. and Wang, L. (2016). The evolution of a generalized neural learning rule. In *Neural Networks (IJCNN), 2016 International Joint Conference on*, pages 4688–4694. IEEE.

Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., and Wermter, S. (2019). Continual lifelong learning with neural networks: A review. *Neural Networks*.

Risi, S. and Stanley, K. O. (2010). Indirectly encoding neural plasticity as a pattern of local rules. In *International Conference on Simulation of Adaptive Behavior*, pages 533–543. Springer.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533.

Rumelhart, D. E., McClelland, J. L., Group, P. R., et al. (1987). *Parallel distributed processing*, volume 1. MIT press Cambridge, MA.

Runarsson, T. P. and Jonsson, M. T. (2000). Evolution and design of distributed learning rules. In *Combinations of Evolutionary Computation and Neural Networks, 2000 IEEE Symposium on*, pages 59–63. IEEE.

Sejnowski, T. J. and Tesauro, G. (1989). The Hebb rule for synaptic plasticity: algorithms and implementations. In *Neural models of plasticity*, pages 94–103. Elsevier.

Sipper, M., Sanchez, E., Mange, D., Tomassini, M., Pérez-Uribe, A., and Stauffer, A. (1997). A phylogenetic, ontogenetic, and epigenetic view of bio-inspired hardware systems. *IEEE Transactions on Evolutionary Computation*, 1(1):83–97.

Soltoggio, A., Bullinaria, J. A., Mattiussi, C., Dürr, P., and Floreano, D. (2008). Evolutionary advantages of neuromodulated plasticity in dynamic, reward-based scenarios. In *Proceedings of the 11th international conference on artificial life (Alife XI)*, pages 569–576. MIT Press.

Soltoggio, A. and Stanley, K. O. (2012). From modulated Hebbian plasticity to simple behavior learning through noise and weight saturation. *Neural Networks*, 34:28–41.

Soltoggio, A., Stanley, K. O., and Risi, S. (2018). Born to learn: The inspiration, progress, and future of evolved plastic artificial neural networks. *Neural Networks*.

Stanley, K. O. (2007). Compositional pattern producing networks: A novel abstraction of development. *Genetic programming and evolvable machines*, 8(2):131–162.

Tonelli, P. and Mouret, J.-B. (2013). On the relationships between generative encodings, regularity, and learning abilities when evolving plastic artificial neural networks. *PloS one*, 8(11):e79138.

Vasilkoski, Z., Ames, H., Chandler, B., Gorchetchnikov, A., Léveillé, J., Livitz, G., Mingolla, E., and Versace, M. (2011). Review of stability properties of neural plasticity rules for implementation on memristive neuromorphic hardware. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 2563–2569. IEEE.

Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics bulletin*, 1(6):80–83.

Yaman, A., Iacca, G., Mocanu, D. C., Fletcher, G., and Pechenizkiy, M. (2019). Learning with delayed synaptic plasticity. In *Proceedings of the Genetic and Evolutionary Computation Conference (Accepted)*. ACM.

Yaman, A., Mocanu, D. C., Iacca, G., Fletcher, G., and Pechenizkiy, M. (2018). Limited evaluation cooperative co-evolutionary differential evolution for large-scale neuroevolution. In *Genetic and Evolutionary Computation Conference, 15-19 July 2018, Kyoto, Japan*.

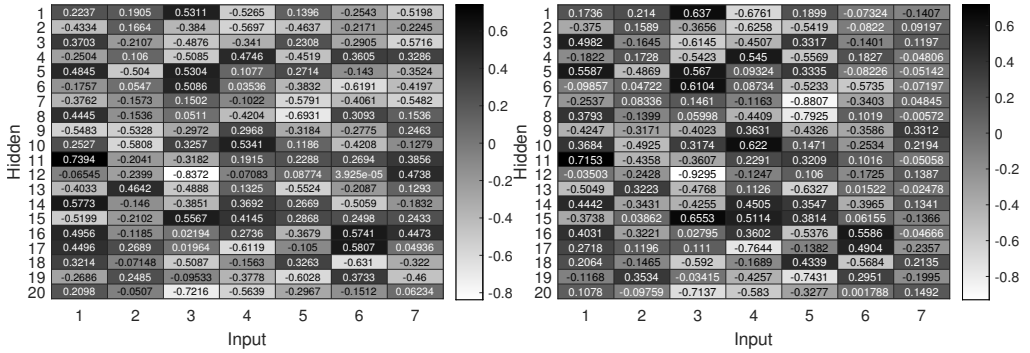
Yao, X. (1999). Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447.

A Change of the synaptic weights

Figures 9 and 10 show the actual values of the connection weights between input and hidden, and between hidden and output layers, in a matrix form. Each column and row in the figures correspond to a neuron in the input, hidden and output layers. In particular, Figures 9a and 10a show the initial connection weights between input and hidden, and between hidden and output layers, sampled randomly. We performed three seasonal changes in the following order: summer, winter, summer. We used the

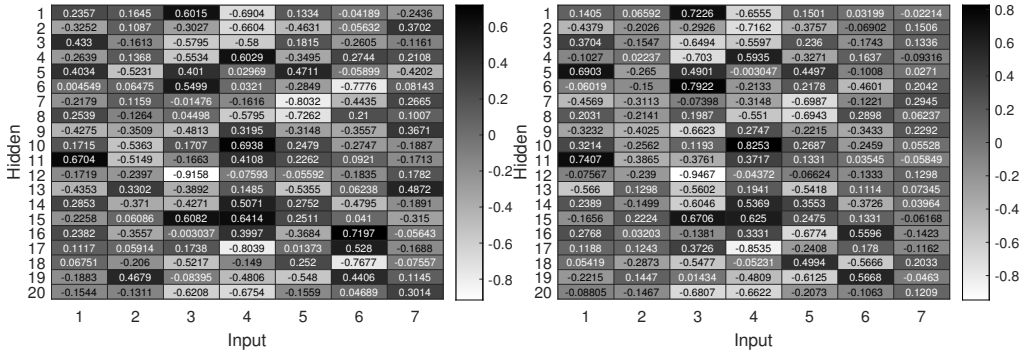
best performing evolved plasticity rule to perform synaptic updates during the seasons. We provide the rest of the figures to show the connection weights after each consecutive season.

The connection weights between input and hidden layers appear to be distributed in the range [-1,1]; on the other hand, the connection weights between hidden and output layers tend to be distributed within the range [0,1] at the end of each season. This may be due to the activation function of the output neurons, where only the neuron with the maximum activation value is allowed to fire.



(a) Randomly sampled initial connection weights.

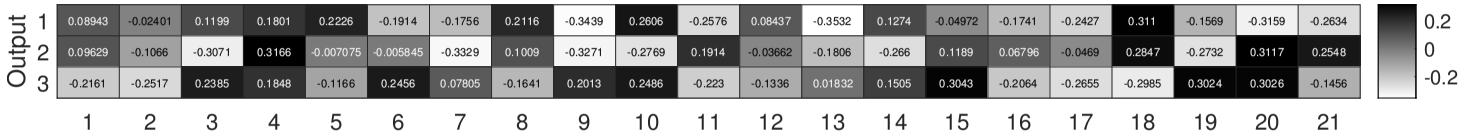
(b) Connection weights after the first summer season.



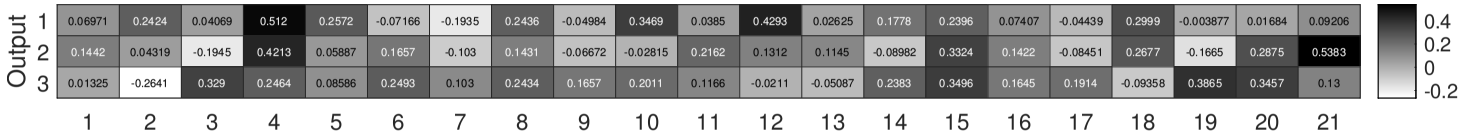
(c) Connection weights after the first winter season.

(d) Connection weights after the second summer season.

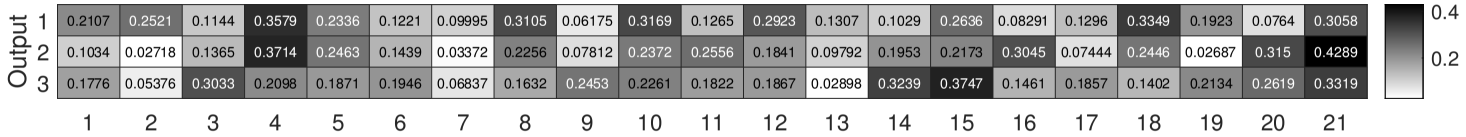
Figure 9: Heat map of the intensities of the connection weights between the input and hidden neurons during a single run using the best evolved plasticity rule. The x and y-axes show the input and hidden neuron indices respectively (7th column shows the biases). Each connection on the heat map is color coded based on its intensity from -1 to 1. Figure 9a shows the initial state of the connection weights that are assigned randomly. Figures 9b, 9c and 9d show the weights after summer, winter and summer seasons.



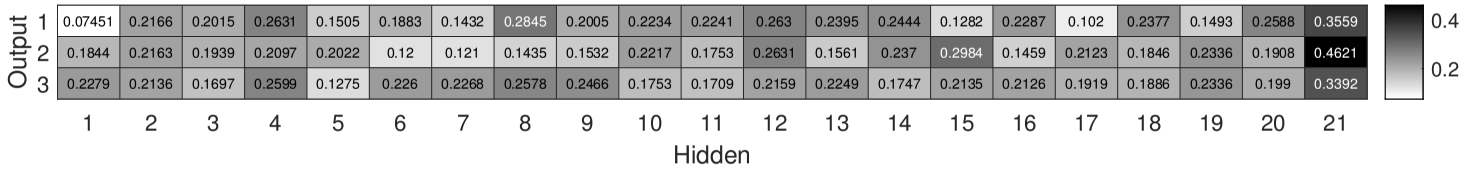
(a) Connection weights after the initial random sampling.



(b) Connection weights after the first summer season.



(c) Connection weights after the first winter season.



(d) Connection weights after the second summer season

Figure 10: Heat map of the intensities of the connection weights between the hidden and output neurons during a single run using the best evolved plasticity rule. The x and y-axes show the hidden and output neuron indices respectively (21st column shows the biases). Each connection on the heat map is color coded based on its intensity from -1 to 1. Figure 10a shows the initial state of the connection weights that are assigned randomly. Figures 10b, 10c and 10d show the weights after summer, winter and summer seasons.