Behavioral Plasticity Through the Modulation of Switch Neurons

Vassilis Vassiliades, Chris Christodoulou

Department of Computer Science, University of Cyprus, 1678 Nicosia, Cyprus

Abstract

A central question in artificial intelligence is how to design agents capable of switching between different behaviors in response to environmental changes. Taking inspiration from neuroscience, we address this problem by utilizing artificial neural networks (NNs) as agent controllers, and mechanisms such as neuromodulation and synaptic gating. The novel aspect of this work is the introduction of a type of artificial neuron we call "switch neuron". A switch neuron regulates the flow of information in NNs by selectively gating all but one of its incoming synaptic connections, effectively allowing only one signal to propagate forward. The allowed connection is determined by the switch neuron's level of modulatory activation which is affected by modulatory signals, such as signals that encode some information about the reward received by the agent. An important aspect of the switch neuron is that it can be used in appropriate "switch modules" in order to modulate other switch neurons. As we show, the introduction of the switch modules enables the creation of sequences of gating events. This is achieved through the design of a modulatory pathway capable of exploring in a principled manner all permutations of the connections arriving on the switch neurons. We test the model by presenting appropriate architectures in nonstationary binary association problems and T-maze tasks. The results show that for all tasks, the switch neuron architectures generate optimal adaptive behaviors, providing evidence that the switch neuron model could be a valuable tool in simulations where behavioral plasticity is required.

Keywords:

Email addresses: v.vassiliades@cs.ucy.ac.cy (Vassilis Vassiliades), cchrist@cs.ucy.ac.cy (Chris Christodoulou)

To appear in Neural Networks, doi: 10.1016/j.neunet.2015.11.001 (accepted, in press)

switch neuron, behavioral plasticity, neuromodulation, gating, adaptive behavior, reinforcement learning

1. Introduction

Adaptive organisms have the remarkable ability of adjusting their behavior in response to changes in their environment. Such *behavioral plasticity* is believed to be linked with modifications in the neural circuitry that produces the behavior. These modifications are likely to be caused by mechanisms that go beyond the classical neurotransmission (of excitation or inhibition), such as *neural plasticity* (Churchland and Sejnowski, 1992; Binder et al., 2009) and *neuromodulation* (Katz, 1999). Neural plasticity refers to the capacity of neural circuits for functional or organizational modifications due to previous activity or damage (Churchland and Sejnowski, 1992; Binder et al., 2009). For example, synaptic plasticity, i.e., the strengthening or weakening of synapses, is a major process that underlies learning and memory (Martin et al., 2000) and has been validated through neural recordings (see for example Kandel and Tauc, 1965; Bliss and Lømo, 1973; Markram et al., 1997; Bi and Poo, 1998). Neuromodulation refers to the process where a small number of neurons can influence (modulate) the intrinsic properties of multiple synapses or neurons, through the diffusion of certain neurotransmitters known as neuromodulators (Katz, 1999; Marder and Thirumalai, 2002; Binder et al., 2009). Neuromodulation and neural plasticity can be complementary. For example, the neuromodulator dopamine is believed to play a major role in operant conditioning (Thorndike, 1911; Skinner, 1938) as it was found to encode a reward prediction error (RPE; see for example Schultz, 1998) analogous to temporal difference (TD) error in reinforcement learning (RL, Sutton and Barto, 1998).

While neuromodulation can be used to *gate* plasticity and synaptic transmission, a growing number of studies provide evidence that supports the existence of other types of *synaptic gating* mechanisms, capable of regulating information flow between various sets of neurons (see Gisiger and Boukadoum, 2011, and references therein). Such mechanisms should not be thought of as simply interrupting information flow, as they can also act as permissive gates (Katz, 2003). For example, certain neurons from an area of the brain called the nucleus accumbens (NAcc) were found to implement this type of gating. These NAcc neurons are *bistable*, meaning that they exhibit oscillations between two discrete states: an "up" state (where the membrane

potential is depolarized) during which the neuron generates action potentials (spikes), and a resting, "down" state (where the membrane potential is hyperpolarized) during which the production of action potentials is absent (O'Donnell and Grace, 1995; Grace, 2000). They were found to be part of a gating mechanism that controls whether information from the prefrontal cortex (PFC) is allowed to pass through to the ventral pallidum and further, to the thalamus. More specifically, input from PFC neurons arrives at NAcc neurons, but only those that are in their up state allow the input to propagate forward. What modulates the state of the NAcc neurons is an extra input from the hippocampus (HPC). That is, only the NAcc neurons that are stimulated by HPC neurons enter their depolarized state and subsequently, fire upon receiving input from the PFC (Grace, 2000). For this reason, these neurons are said to implement a type of AND gate, since they fire only if they receive input from both the PFC and the HPC (Gisiger and Boukadoum, 2011). Various other logic gates, such as NOT, Switch, XOR, and Flip-Flop, can be implemented by neural circuits, as demonstrated by Vogels and Abbott (2005).

Apart from bistable neurons, which were found to be abundant in the cortex, other gating mechanisms have been observed in theoretical or experimental data, that feature inhibitory neurons or even oscillations (for examples see Anderson and Van Essen, 1987; Olshausen et al., 1993; Burchell et al., 1998; Floresco and Grace, 2003; Barbas and Zikopoulos, 2007). In all observations and models, certain "gatekeeper" circuits influence synaptic transmission (Gisiger and Boukadoum, 2011). As mentioned above, it has been observed that for NAcc neurons the gatekeepers originate in the HPC. For cortex neurons, experimental evidence suggests that the gatekeepers could originate in the cortex, thalamus and basal ganglia (Gisiger and Boukadoum, 2011). Gisiger and Boukadoum (2011) present a theoretical model where a copy of the gating signal produced by one gatekeeper circuit, can be fed as an input to another gatekeeper circuit. A key observation in that model is the existence of two types of neural pathways: the first implements normal information processing, whereas the second is formed by the gating mechanisms. They hypothesize that such interacting gating circuits could create sequences of gating events that are responsible for the production of structured behavior.

Gruber et al. (2009) used multichannel recordings to investigate rats during spatial exploration of an operant chamber, and during reward-seeking afterwards in the same chamber. They observed that during spatial explo-

ration, the activities of neurons in the NAcc core, i.e., the inner part of the NAcc which is located within the basal ganglia (Gerfen and Wilson, 1996, Chap. 2, p. 372), synchronized with the activity of HPC neurons; however, during reward-seeking, they instead synchronized with the activity of PFC neurons. This suggested that the NAcc core can dynamically select its inputs according to environmental requirements, as it is able to switch its synchronization in a task-dependent manner (Gruber et al., 2009). It has to be noted that the basal ganglia is the structure believed to be associated with action selection (Redgrave et al., 1999) and RL (Barto, 1995; Montague et al., 1996; Schultz et al., 1997). For example, Redgrave et al. (1999) proposed that the action selection problem of vertebrates is solved by a central "switching" mechanism that resides in the basal ganglia. It has also been recently suggested that the release of a peptide called "substance P" in the striatum (i.e., the primary input nucleus to the basal ganglia), allows for rapid switching between actions in action sequences (Buxton et al., 2015).

Gating mechanisms can be seen as implementing a type of "on-off" switch by allowing or interrupting communication between brain regions. Inspired by these gating phenomena in the brain, which seem to play a significant role in various processes such as working memory (Williams and Goldman-Rakic, 1995), attention (Usher et al., 1999), and decision making (Cisek et al., 2009), we ask whether we could design an abstract computational model that can be used for the purpose of adaptive behavior. For this reason, we adopt the artificial intelligence agenda of rational decision making (Russell and Norvig, 2003), where an agent tries to maximize its reward intake (Sutton and Barto, 1998). The agent is controlled by artificial neural networks (NNs), as they are very well suited for simulating adaptive behavior, due to the possibility of implementing memory (through recurrent connections), and learning (through plasticity rules). In this paper, we do not focus on learning behavior per se, but rather on behavior exploration. More specifically, a central hypothesis of this work is that once some general neural circuits are established for certain behaviors through possibly neural plasticity mechanisms (or other methods), neuromodulation alone can be used to switch these behaviors by selectively gating various pathways accordingly.

We implement such a gating mechanism by introducing a novel type of an artificial neuron we call *"switch neuron"* that can be used in NNs. Instead of implementing an on-off switch for certain connections, this unit selects which one of its incoming connections is allowed to propagate its signal forward, by opening its gate while closing the gate of all others. The role of the switch

neuron is to endow an agent with different behaviors and the ability to flexibly switch them as needed. The switching activity is controlled by modulatory signals that encode some information about the reward received by the agent. In order to create sequences of gating events and structured exploration, we additionally introduce a way for switch neurons to modulate other switch neurons. This is done by placing them in appropriate switch modules. We assess our model by designing appropriate switch neuron architectures for nonstationary association tasks (Section 3.1) and discrete T-maze problems (Section 3.2). We show in all tasks that these architectures perform optimal deterministic exploration when the goal changes, therefore, illustrating that our approach advances the field of NNs by creating more adaptive networks.

Note that the switch neurons of this paper should not be viewed in a strict biological sense, but rather in a functional sense. They are inspired by biological phenomena, but they are artificially constructed to perform certain computations. Thus, throughout this study, we use the word "neuron" for the switch neuron, but note that this is a purely artificial unit. In other words, despite the strong biological inspiration for the design of the switch neuron model, our paper does not contribute to any advances in biological areas. If mechanisms similar to the switch neuron model exist in the brain, they could either be in the form of individual cells, population of cells, or groups of interconnected neurons.

The remainder of the paper is organized as follows. Section 2 describes our approach by introducing the switch neuron and switch module. These are integrated in NN architectures designed specifically for the experiments reported in Section 3 along with the results. Section 4 discusses our results and directions for future work, and the conclusions are given in Section 5.

2. Approach

2.1. Artificial neurons

The usual formulation of an artificial neuron involves the integration of incoming signals $\mathbf{y} := (y_1, y_2, ..., y_N)$ and parameters $\mathbf{w} := (w_1, w_2, ..., w_N)$ through an accumulation or integration function $G(.)$ resulting in the neuron's activity $a(t) := G(\mathbf{y}, \mathbf{w})$ at time t. This activity is then fed through an activation function $F(.)$ resulting in the neuron's output $y(t) := F(a(t))$.

In the work of Soltoggio et al. (2008), the distinction between standard neurons and modulatory neurons is made. Standard neurons can be modeled as above, with their output interpreted as a "standard signal". Modulatory

neurons, on the other hand, transform the incoming standard signals by emitting special "modulatory signals" that affect the synaptic plasticity of a target neuron. A critical modeling aspect is that both types of neurons have an internal value for a standard activation $a^{(std)}(t)$ and a modulatory activation $a^{(mod)}(t)$. In this work, we adopt a more flexible formulation where connections instead of neurons can either be standard or modulatory. This way the same neuron can emit both a standard and a modulatory signal if needed. This change is important for the modeling of switch modules, as it will be described in Section 2.4.

Before introducing the switch neuron model, we start with a more general formulation of the neurons that are used in this work. An active neuron $\mathbf{n}_i$ (i.e., hidden or output) consists of two parts that are responsible for the calculation and storage of its standard activation and modulatory activation:

$$\mathbf{n}_i := \langle \mathbf{s}_i^{(std)}, \mathbf{s}_i^{(mod)} \rangle \tag{1}$$

$\mathbf{s}_i^{(std)}$ and $\mathbf{s}_i^{(mod)}$ are tuples that hold the parameters for computing and storing the standard output and the modulatory output of the neuron respectively, and are represented as

$$\mathbf{s}_i^{(x)} := \langle F_i^{(x)}(.), G_i^{(x)}(.), a_i^{(x)}(t), y_i^{(x)}(t), \mathbf{z}_i^{(x)}, \mathbf{w}_i^{(x)}, \mathbf{d}_i^{(x)} \rangle \tag{2}$$

where for part (x) (this can be the standard or modulatory part) each component is described below by omitting the (x) index for simplicity: $F_i(.)$ is the activation function, $G_i(.)$ is the integration function, $a_i(t) := G_i(.)$ is the activity at time t, $y_i(t) := F_i(a_i(t))$ is the output at time t, $\mathbf{w}_i := [w_{1i}, w_{2i}, ..., w_{Ni}]^T$ is a vector that contains the weights of the presynaptic connections, $\mathbf{d}_i := [d_{1i}, d_{2i}, ..., d_{Ni}]^T$ is a vector that contains the delays of the presynaptic connections where N is the number of presynaptic connections and $d_{ji} \in \{0, 1, 2, ..., d_{max}\}$ with d_{max} being the maximum delay a connection could have, and $\mathbf{z}_i := [y_i(t-1), ..., y_i(t - max(\forall j d_{ij}))]^T$ is a vector that contains the previous outputs of neuron i, with $max(\forall j d_{ij})$ being the maximum delay from all *outgoing* connections of neuron i. Note that in the case of a feedforward connection, the delay d_{ji} takes the value of zero. Also note that in the case of feedforward networks the vector $\mathbf{z}_i$ does not exist since there is no need to store the previous outputs, and in the case of normal recurrent networks, where the delay of all connections is equal to one, the vector $\mathbf{z}_i$ contains only one element, i.e., $y_i(t-1)$. As it will be shown in Section 3.2 the connection delays can be useful for disambiguating the state

in partially observable problems. Various models can be implemented by substituting different integration and activation functions for both the standard and modulatory part. The activity at time t (standard or modulatory accordingly) is computed as

$$a_i(t) := G_i(\mathbf{y}(\mathbf{t} - \mathbf{d}_i), \mathbf{w}_i) \tag{3}$$

where $\mathbf{y}(\mathbf{t} - \mathbf{d}_i) := [y_1(t - d_{1i}), y_2(t - d_{2i}), ..., y_N(t - d_{Ni})]^T$ is a vector that contains the outputs of the presynaptic neurons at times $\mathbf{t} - \mathbf{d}_i$.

In the experiments we present in this work (Section 3), in all neurons apart from the switch neurons: (i) the modulatory integration function is the "weighted-sum", i.e., $a_i^{(mod)}(t) = \sum_{w_{ji} \in Mod} w_{ji} \cdot y_j^{(std)}(t - d_{ji})$ (where Mod is the set of incoming modulatory connections), and (ii) the modulatory activation function is the "hyperbolic tangent", i.e., $y_i^{(mod)}(t) = tanh(a_i^{(mod)}(t))$. Note that although we use these functions based on the work by Soltoggio et al. (2008), in our experiments the modulatory parts of all neurons except for the switch neurons do not affect the computation of the NN architectures, since modulatory connections arrive only at switch neurons (see Section 3); therefore, any function could be used without a change in the outcome.

2.2. The switch neuron model

The rationale behind our switch neuron model is that the incoming signals are not integrated, but instead only one is allowed to be propagated forward, while all the others are blocked. The switching activity of the neuron is influenced by modulatory signals. More specifically, modulatory signals change the level of modulatory activity of the switch neuron and the level of modulatory activity determines from which of its incoming connections the signal is allowed to be propagated forward. The functional role of the switch neuron is to endow the agent with different behaviors by enabling information flow through different parts of the network.

A switch neuron first computes its modulatory part and then its standard part. Formally, its standard part uses the "linear" activation function $y_i^{(std)}(t) = a_i^{(std)}(t)$, while the integration function calculates its standard activation as

$$a_i^{(std)}(t) = w_{ji} \cdot y_j^{(std)}(t - d_{ji}) \tag{4}$$

where $a_i^{(std)}(t)$ is the standard activation of the switch neuron at time t, $y_j^{(std)}(t)$ is the standard output of the jth presynaptic node at time t and

w_{ji} is the weight of the (standard) connection. The jth index is selected according to:

$$j = \left\{ k \quad \middle| \quad \frac{k-1}{n} \leq y_i^{(mod)}(t) < \frac{k}{n}, \quad \forall k = 1, 2, ..., n \right\} = \lfloor n \cdot y_i^{(mod)}(t) \rfloor \quad (5)$$

where $n \in \mathbb{N}^+$ is the number of incoming standard connections and $y_i^{(mod)}(t) \in [0, 1)$ is the modulatory output of switch neuron i at time t. This equation effectively partitions the range of modulatory output $[0, 1)$ into n homogeneous intervals, with the size of each interval being equal to $1/n$. It also implies an 'order relation' between the connection indices, i.e., connection k comes after connection $k-1$, meaning that if currently connection $k-1$ is selected and the next modulatory output is at most $1/n$ greater than the current, then the next selected connection will be connection k. Note that the level of modulatory activity can change this index and consequently the function of the switch neuron. Therefore, the switch neuron could be viewed as possessing a type of *intrinsic plasticity* (Triesch, 2007). This is also consistent with work showing that neuromodulation can alter the intrinsic properties of neurons (see Marder and Thirumalai, 2002, and references therein).

An important design decision is that the modulatory signals do not directly decide the selected connection of the switch neuron, since such an approach might have been difficult to control. Instead, they do so indirectly by determining the *change* in the modulatory activity of the switch neuron. Concretely, the modulatory part of the switch neuron uses the linear activation function $y_i^{(mod)}(t) = a_i^{(mod)}(t)$ and an integration function that acts as a perfect integrator of the weighted-sum of the incoming modulatory signals

$$a_i^{(mod)}(t) = a_i^{(mod)}(t-1) + \sum_{w_{ji} \in Mod} w_{ji} \cdot y_j^{(std)}(t - d_{ji}) \quad (6)$$

but ensures that the activity is kept in the range $[0, 1)$ using

$$a_i^{(mod)}(t) = frac(a_i^{(mod)}(t)) = a_i^{(mod)}(t) - \lfloor a_i^{(mod)}(t) \rfloor \quad (7)$$

It is worth noting that the initial value of the modulatory activity is set to $a_i^{(mod)}(0) = 1/(2n)$, i.e., in the "middle" of the first interval. Although in this work we do not consider any randomness in the modulatory signals, by setting the modulatory activation in the middle of an interval, we provide some robustness to possibly noisy modulatory signals.

Equation 7 can be seen as connecting the maximum with the minimum of the range $[0, 1)$ by stripping away the integer part of $a_i^{(mod)}(t)$. This effectively implements a 'cyclic' relation in the space of indices, a biological evidence of which is still to be found. This means that if connection n (i.e., the last one) is currently selected and the next modulatory output is at most $1/n$ *greater* than the current, then the next selected connection will be connection 1 (i.e., the first one). Similarly, if connection 1 is currently selected and the next modulatory output is at most $1/n$ *less* than the current, then the next selected connection will be connection n (see Figure 1).

2.3. Modulatory signal

The equations of the modulatory integration function of the switch neuron show that both positive and negative modulatory signals can explore the connection indices. In particular, if the modulatory signal has the value of +1 or -1 (and scaled by a weight of $1/n$), this is interpreted as an 'instruction' for the switch neuron to select the *next* index in a "clockwise" or "counter-clockwise" manner respectively (see Figure 1). A signal with magnitude less than ± 1 does not ensure this, whereas a signal with magnitude greater than ± 1 would skip at least an index (depending on n). Therefore, the intensity of the signal (coupled with the weight) determines the index to be selected. Note that since the modulatory integration function of the switch neuron behaves as a perfect integrator, the modulatory signals could accumulate before switching to the next index. That is, switching could occur after receiving multiple modulatory signals.

What described above shows that the modulatory signals of switch neurons differ from modulatory signals devised for synaptic plasticity rules (see for example, Soltoggio and Stanley, 2012), that resemble a RPE, in the sense that there is no reinforcing of pathways when a positive signal is used. A zero signal indicates that the current behavior must not change. However, both positive and negative signals elicit a change in behavior, i.e., an exploratory action (if the NN architecture is appropriate) if they manage to significantly alter the modulatory activity of a switch neuron.

2.4. The switch module: a module of three neurons

In order to enable the creation of sequences of gating events, which could potentially confer benefits in sequential decision making problems, there needs to be a way for switch neurons not only to be modulated, but also to be able to modulate other switch neurons. This, however, implies that

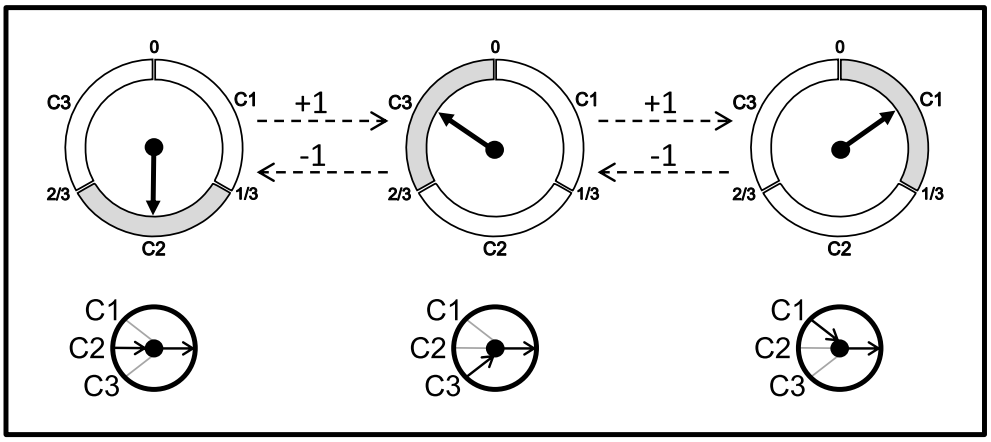


Figure 1: Modulatory "wheel". The wheel (*top*) shows how the level of modulatory activation of the switch neuron (*bottom*) affects its decision. The switch neuron has 3 incoming connections, thus, it partitions the range $[0, 1)$ into 3 homogeneous intervals of size $1/3$, one for each incoming connection. Initially, connection 2 (C2) is selected as the modulatory activation falls in the range $[1/3, 2/3)$. Upon the reception of a positive signal (+1), which is multiplied by the weight $1/3$, the modulatory activation falls in the range $[2/3, 1)$, thus, connection 3 (C3) is selected. Another positive signal makes the modulatory activation fall in the range $[0, 1/3)$, thus, selecting connection 1 (C1). Positive modulatory signals have this clockwise behavior, whereas negative ones behave similarly in a counterclockwise manner.

the switch neurons 'need' to emit two *distinct* signals, a standard and a modulatory one, with both being calculated by *different* functions. It is important to note that this is not equivalent to having a neuron calculating its (standard) output and using two outgoing connections from it, a standard and a modulatory one, since both connections would carry information about the *same* signal. This stems from the fact that artificial neurons are traditionally designed to output a *scalar* value, i.e., their standard output, $y_i^{(std)}(t)$, and not a vector of values. That is, the modulatory output of the neurons, i.e., $y_i^{(mod)}(t)$, is *not* transmitted to other neurons, and in previous work it has been used locally to adjust the strength of synaptic plasticity (Soltoggio et al., 2007, 2008; Soltoggio, 2008; Dürr et al., 2008, 2010; Soltoggio and Jones, 2009; Risi et al., 2009, 2010; Risi and Stanley, 2012; Sher, 2012; Silva et al., 2012; Arnold, 2011; Arnold et al., 2012, 2013; Tonelli and Mouret, 2011b,a, 2013; Nogueira et al., 2013; Ellefsen, 2013; Coleman et al., 2014; Lehman and Miikkulainen, 2014; Mouret and Tonelli, 2014; Yoder and Yaeger, 2014). Therefore, in order to ensure the simplicity of implementation

and to make the model more widely applicable, it was deemed necessary to make the following change: a switch neuron is replaced with a *module* of three neurons (described below), only when interactions between switch neurons need to be modeled. More specifically, the switch neuron that 'needs to emit' its own modulatory signal is replaced with this three-neuron module, which we refer to as a "switch module" throughout this paper.

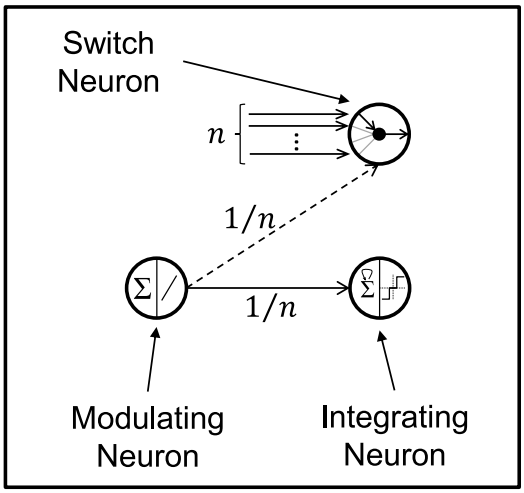


Figure 2: The switch module contains three neurons. The "switch neuron" has n incoming standard connections (shown by *solid* lines) and can be seen as endowing an agent with different behaviors. The "modulating neuron" is responsible for altering the level of modulatory activation of the switch neuron and for this reason it connects to the switch neuron with a modulatory connection (shown by a *dashed* line). The "integrating neuron" integrates the modulatory signals emitted from the modulating neuron, using a standard connection that has the same weight as the modulatory one, which is equal to $1/n$, and fires when its activation exceeds a threshold value; this neuron is responsible for connecting different switch modules/neurons.

The first neuron in the switch module is the *switch neuron* and is described in Section 2.2. The second neuron is responsible for altering the level of modulatory activity of the switch neuron, therefore, being referred to as "*modulating neuron*". The role of the modulating neuron is to alter the behavior of the agent when needed. Finally, the third neuron is responsible for integrating a copy of the signal emitted by the modulating neuron. It outputs a signal only when a certain threshold is reached; it then immediately resets to its initial state. The role of the third neuron is for connecting different switch modules/neurons, effectively providing a way to modulate other

switch neurons. This third neuron is referred to as "*integrating neuron*". The switch module is illustrated in Figure 2 where we show how the three neurons connect with each other, while in Figure 3 (right box), we show how the switch modules can be used to modulate one another. We now proceed to a detailed description of the modulating and integrating neurons.

2.4.1. The modulating neuron

The modulating neuron uses the weighted-sum integration function and the linear activation function for calculating its standard activation and standard output respectively. Therefore, its (standard) output is computed as

$$y_i^{(std)}(t) = a_i^{(std)}(t) = \sum_{w_{ji} \in Std} w_{ji} \cdot y_j^{(std)}(t - d_{ji}) \tag{8}$$

where Std is the set of incoming standard connections. The modulating neuron is connected to the switch neuron via a *modulatory* connection and to the integrating neuron via a *standard* connection, both having a *shared weight* value equal to $1/n$, where n is the number of incoming connections of the switch neuron (see Figure 2).

This special connectivity was designed to permit the following two desired characteristics. The first is related to the modulation of an individual switch neuron and the exploration of its states; it is achieved by the modulatory connection from the modulating neuron. The second characteristic is related to interactions between switch neurons. By feeding the (weighted) signal of the modulating neuron to the integrating neuron, the latter becomes able to 'know' when all the states of the switch neuron (i.e., n connections) have been explored. It could then emit a signal to other parts of the network.

2.4.2. The integrating neuron

The integrating neuron integrates the signals received by the modulating neuron and fires when some threshold is reached. The integration function is a perfect integrator of the weighted-sum of the incoming standard signals:

$$a_i^{(std)}(t) = a_i^{(std)}(t-1) + \sum_{w_{ji} \in Std} w_{ji} \cdot y_j^{(std)}(t - d_{ji}) \tag{9}$$

The activation function is the following:

$$y_i^{(std)}(t) = \begin{cases} 1 & \text{if } a_i^{(std)}(t) \geq \theta \\ -1 & \text{if } a_i^{(std)}(t) < -\theta \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

where $\theta > 0$ is a (positive) threshold value that is set to 1 in all experiments. This neuron resembles integrate-and-fire neurons (Lapicque, 1907; Tuckwell, 1988), but has a symmetric form in the sense that the output can be both positive or negative. When the output is not zero, i.e., 1 or -1, the activation of the neuron is reset to a baseline value, b, which is set to 0 in all experiments of this study.

$$a_i^{(std)}(t) = \begin{cases} 0 & \text{if } y_i^{(std)}(t) = 1 \quad or \quad y_i^{(std)}(t) = -1 \\ a_i^{(std)}(t) & \text{otherwise} \end{cases} \tag{11}$$

2.5. Figure simplification

In order to simplify the figures in the sections that follow, we present (Figure 3) a depiction of how a connected subnetwork (i.e., a network that is part of a larger NN) of switch neurons is illustrated (in the figures of this paper) and how it can be actually implemented (conceptually and in source code). *Solid* lines represent standard connections and *dashed* lines represent modulatory connections. Whenever we illustrate (in a figure) that there is

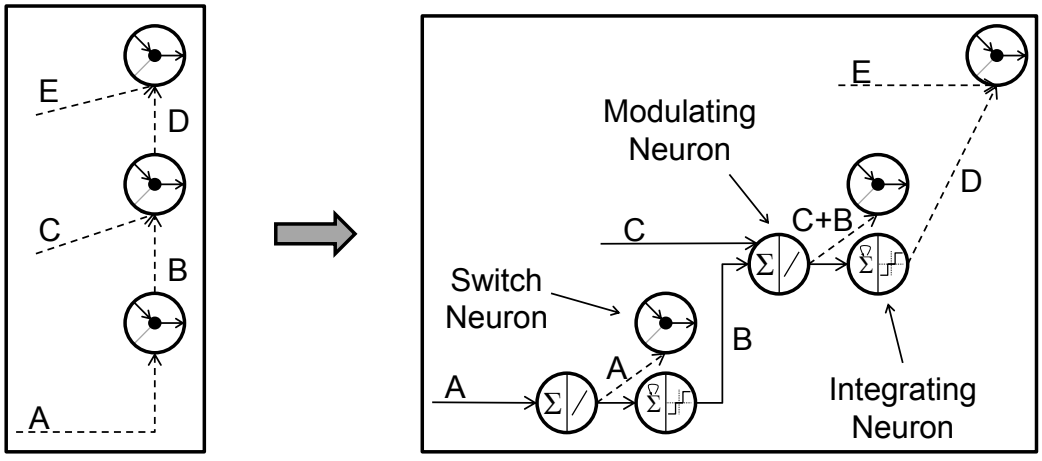


Figure 3: Illustration and implementation of switch neurons and switch modules. The left box shows how a network of switch neurons is illustrated throughout this work. The right box shows how this network can be implemented. *Solid* lines represent standard connections and *dashed* lines represent modulatory connections. Whenever a modulatory connection is shown to be flowing out from the top of a switch neuron (see left box) means that a switch module needs to be used (see right box). See text for more details.

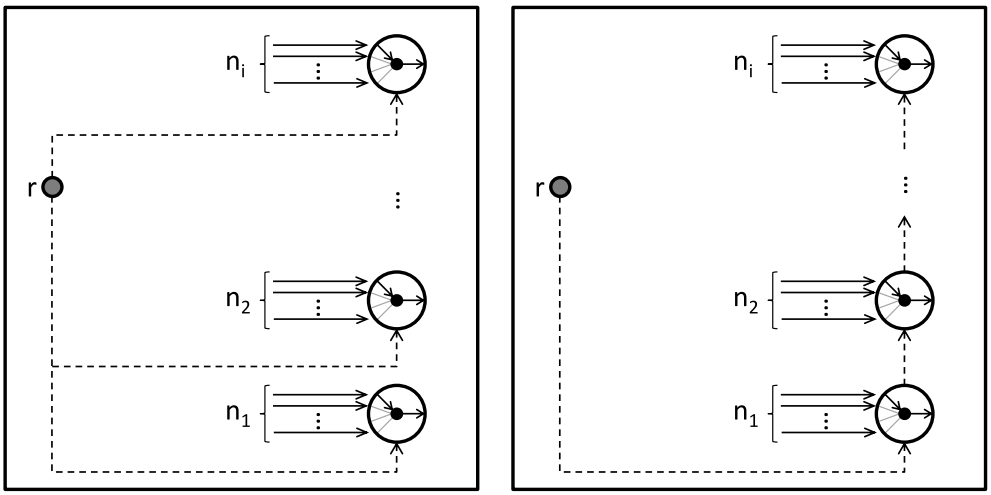
a modulatory connection flowing *out* (e.g., from the top or bottom) of a

switch neuron (see the example in Figure 3, left box), then that switch neuron is substituted (in the implementation) with the switch module described in Section 2.4 as shown in Figure 3. In the example of Figure 3, we illustrate some external signals A and C, and an internal signal B as modulatory signals, however, we implement them as standard signals. This is due to the fact that the switch neuron, on which they are connected, interacts with another switch neuron via an outward connection. Signals E and D are both illustrated and implemented as modulatory signals without using any switch module. This is because their connecting switch neuron does not emit any modulatory signal, so not using the switch module slightly reduces the complexity of the network. Figure 3 illustrates that each modulating neuron has the role of accumulating all incoming modulation, converting it into a modulatory signal that is fed into the switch neuron, and propagating it forward as a standard signal that is fed into the integrating neuron.

2.6. Modulatory behavior

In this study, we use two modulatory topologies that affect the modulatory behavior of the network in a different manner. The first topology enables parallel/simultaneous modulation of multiple switch neurons by an external signal that is roughly related to the reward obtained by the agent (or a RPE signal). This is shown in Figure 4a, where solid lines represent standard connections and dashed lines represent modulatory connections. Unless an individual modulatory connection is gated, a continuous stream of non-zero reward signals modulates all switch neurons at every time step.

The second topology, shown in Figure 4b, organizes the switch neurons one after the other along a *modulatory pathway*. The result is that switch neurons can modulate other switch neurons sequentially. That is, in this topology, modulation does not only come from an external source, but it can also be calculated internally by the network. This is motivated and inspired by studies on intrinsic neuromodulation (for example, see Katz and Frost, 1996), as well as what Gisiger and Boukadoum (2011) call "*the second pathway*" that "*could play various roles*". In order to understand how often each neuron on this pathway is modulated, suppose that Switch_1 has n_1 incoming connections, Switch_2 has n_2 incoming connections,..., Switch_i has n_i connections. External modulation modulates Switch_1, Switch_1 modulates Switch_2, Switch_2 modulates Switch_3,..., Switch_{i-1} modulates Switch_i. Now assume that a continuous stream of -1 (or +1) external modulation is received. This means that Switch_1 will be modulated at every time step,



(a) Topology for parallel (external-only) modulation.

(b) Topology for sequential (external and internal) modulation.

Figure 4: Modulatory topologies. The architecture in (a) is an example where the external reward signal modulates all switch neurons. As a result, all switches are modulated in parallel at every time step if the reward signal is non-zero. The architecture in (b) is an example where all switches reside on a *modulatory pathway* and thus, they are modulated in sequence: the external reward signal modulates Switch_1, Switch_1 modulates Switch_2, and generally Switch_{i-1} modulates Switch_i. As a result, if the reward signal is always -1 or +1, Switch_1 is modulated at every time step, Switch_2 is modulated every n_1 steps, and generally Switch_i is modulated every $\prod_{j=1}^{i-1} n_j$ steps. *Solid* lines represent standard connections and *dashed* lines represent modulatory connections.

Switch_2 will be modulated every n_1 time steps, Switch_3 will be modulated every $n_1 \times n_2$ time steps, and generally, Switch_i will be modulated every $n_1 \times n_2 \times ... \times n_{i-1} = \prod_{j=1}^{i-1} n_j$ time steps. This implies that this type of modulatory connectivity can explore all permutations of the connections, which are equal to $\prod_{j=1}^{i} n_j$, in an ordered manner. This is useful because a different permutation might mean a different behavior policy for an agent. Note that the standard signals could come from any input, hidden or output neurons. Similarly, the (standard) signal from switch neurons could be fed to other hidden or output neurons.

3. Experiments and results

In the following sections we present our experimental setup, the NN architectures used and the results obtained in two sets of domains. The first

is a broad set of association problems and the second is a set of T-maze domains. Both types of problems involve reward associations that are hidden from the agent and change with time.

3.1. Nonstationary binary association problems

The first set of domains we investigate is association problems. The general setting is to make an agent learn the random associations between binary input patterns and binary output patterns based on feedback that comes in the form of a reward signal. These associations can be re-randomized at certain points in time, requiring from the agent to unlearn the previous associations and re-learn the new ones. It is important to note that the reward signal is a scalar value and not a vector that represents the error of each output variable as it is done in supervised learning / classification settings. For this reason, the agent needs to be able to explore all possible output patterns for each given input pattern. The reward signal is not delayed, therefore, there is no temporal credit assignment problem (Sutton and Barto, 1998).

The number of inputs is n and the number of outputs is m. There are four types of association problems: (i) one-to-one, where each of the n inputs needs to be associated with one of the m outputs, (ii) one-to-many, where each of the n inputs needs to be associated with one of the 2^m possible output patterns, (iii) many-to-one, where each of the 2^n input patterns needs to be associated with one of the m outputs, and (iv) many-to-many, where each of the 2^n input patterns needs to be associated with one of the 2^m possible output patterns. Therefore, in each case, all possible input vectors need to be associated with some output vectors. The number of possible output vectors corresponds to the number of actions available to an agent in a multi-armed bandit setting (Robbins, 1952; Gittins, 1979; Sutton and Barto, 1998). Table 1 shows the number of possible association sets for every type of association problem and the expected number of time steps needed to learn a random association set of each type.

3.1.1. Simulation

The experiments are performed as follows. Initially, all input vectors are randomly associated with an output vector (i.e., action) accordingly. Each input vector is presented sequentially to the network, the agent then selects an action (according to the NN architecture), the action is translated to the corresponding output vector, and the episode ends. If the output

Table 1: Types of association problems, their corresponding number of possible association sets and the (expected) time steps needed to learn a random association set of each type. n is the number of inputs and m is the number of outputs.

Type of association problem	Number of possible association sets	Time needed to learn (in steps)
One-to-One	m^n	$n(m-1)$
One-to-Many	$(2^m)^n$	$n(2^m-1)$
Many-to-One	m^{2^n}	$2^n(m-1)$
Many-to-Many	$(2^m)^{2^n}$	$2^n(2^m-1)$

vector chosen by the agent was correct for the given input, i.e., the input-output pair exists in the association set, then a reward of 0 is provided, otherwise a reward of -1. This reward function was designed to be suitable with the modulatory signals the switch neurons 'understand', i.e., only non-zero modulation changes the behavior. Before feeding the next input to the network, a second activation (forward propagation) is done by keeping the current input active and setting the reward signal appropriately to its corresponding input. This is necessary for allowing the network to adjust its internal state if needed. At that point the new episode begins and the next input is fed to the network. After repeating this procedure for a number of episodes, the association set is randomized and the network needs to adapt by unlearning the previous associations and learning the new ones. This is repeated until a maximum number of episodes is reached.

The experiments are run for 100 independent trials. For all types of problems apart from the many-to-many problems, the number of episodes is set to 2000 with the association sets being randomized every 500 episodes. For the many-to-many problems, these numbers are multiplied by 10, to accommodate the exponentially larger number of associations. In particular, the number of episodes is set to 20,000 and the association sets are randomized every 5000 episodes. Each episode lasts one step and the reported performance is the average reward per episode over all trials.

3.1.2. NN architectures

Figure 5 presents a switch neuron network that optimally solves one-to-one association problems. For simplicity we use $n = m = 3$, however, we should emphasize that it is straightforward to extend the architecture for an

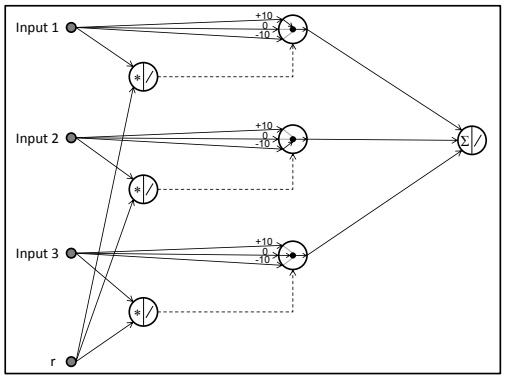
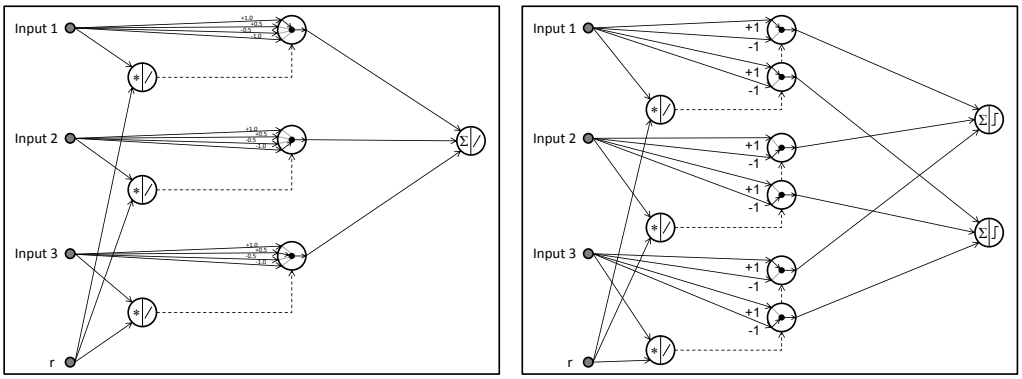


Figure 5: Architecture that solves one-to-one association problems with number of stimuli n and actions m equal to three. *Solid* lines represent standard connections and *dashed* lines represent modulatory connections. The action is decided from the activation of the linear output unit, which is bounded in $[-10, 10]$. The current state of switch neurons associates Input 1 with Action 1, Input 2 with Action 3, and Input 3 with Action 2.

arbitrary number of stimuli/inputs (n) and actions/outputs (m). The NN uses $n + 1$ input neurons (n for all the stimuli, and one for the modulation signal), n switch neurons and product units (one for each input neuron) and a single output neuron. The number of connections is $n(m + 4)$. The m actions are encoded by the linear output neuron. An alternative architecture where m binary output nodes are used to encode the m actions would require a change in the switch neuron model in a way that it gates its outgoing connections instead of its incoming connections. The linear output neuron is bounded in the range $[-10, 10]$. In the example, where $m = 3$, Action 1 is selected if the output is in $(+3.3, +10]$, Action 2 is selected if the output is in $[-3.3, +3.3]$, and Action 3 is selected if the output is in $[-10, -3.3)$. For this reason, the weights take the values of +10 (for selecting Action 1), 0 (for selecting Action 2), and -10 (for selecting Action 3). Note that if a different activation function is used, then these weights could change. The current states of the switch neurons in the figure associate Input 1 with Action 1, Input 2 with Action 3, and Input 3 with Action 2. Note that each input neuron is connected with its corresponding switch neuron using not a single connection, but a number of connections equal to the number of actions, i.e., three. Therefore, if the problem requires more actions then more connections need to be added each one corresponding to each action with an appropriate weight, as discussed above. The role of the product unit is for gating the modulation signal. More specifically, if the reward signal is non-zero, then

modulation is applied to the switch neuron whose input was active. The modulatory weight is equal to $1/m$ in order to be able to cycle through all incoming connections when a -1 modulatory signal is received. All other connections have a weight of 1.0.

Figure 6 illustrates two architectures that optimally solve one-to-many association problems with $n = 3$ inputs and $m = 2$ outputs. The NN architecture in Figure 6a uses the approach followed when designing the NN of Figure 5. More specifically, a single linear neuron is used to encode the 2^m actions. This is done by allowing it to take 2^m different values, which is achieved by having 2^m different connections from each input. The actual weight values of the standard connections that feed to the switch neurons do not matter as long as they are distinct and there is a mapping from each weight to each action. In Figure 6a, this is achieved by bounding the linear unit in the range [-1,1] and splitting it to the $2^m = 2^2 = 4$ weight values to encode the 4 actions, i.e., $+1$ for Action 1, $+0.5$ for Action 2, -0.5 for Action 3 and -1 for Action 4.



(a) Architecture with one output neuron. (b) Architecture with m output neurons.

Figure 6: Architectures that solve one-to-many association problems with $n = 3$ and $m = 2$. The architecture in (a) uses one linear output neuron that encodes the $2^m = 4$ actions by mapping them to 2^m distinct values; these are determined by the connection weights that feed on to the switch neurons, which are +1, +0.5, -0.5 and -1. The architecture in (b) uses m binary output neurons that encode the actions as a bit pattern. All standard connections apart from the ones that feed to the switch neurons have a weight of 1.0. All modulatory connections have a weight of $1/2^m = 0.25$ in (a) and $1/2 = 0.5$ in (b).

In contrast, Figure 6b depicts an architecture where m binary output units are used, instead of just one. The 2^m actions are encoded in the bit pattern of the output units. For example, the bit pattern "01" means that

the first neuron has an output of 0 and the second neuron has an output of 1. In this architecture, the switch neurons need to be able to work together, so as to optimally explore all possible bit patterns. For each input, there are m switch neurons that connect to their corresponding output neuron. Each switch neuron can take the value of $+1$ or -1. This means that the output neurons can either take the value of $+1$ or 0 respectively, as they use the Heaviside activation function. Remember from Section 2.5 that the bottom switch neuron for each input is actually implemented as a module of neurons, since it is the only one that emits a modulatory signal. This means that its integrating neuron will fire, and consequently the upper neuron will be modulated, only when all two states of the bottom neuron are explored (by the -1 modulatory signal). The firing will cause the integrating neuron to reset, and will also force the upper switch neuron to modify its state. This procedure will be repeated until a zero modulatory signal is received. A simple analysis of both architectures (not shown) reveals that it is preferable to use Architecture 2 (Figure 6b) when $m \geq 5$ as it uses significantly fewer connections than Architecture 1.

So far, we have shown how to solve one-to-one and one-to-many problems, therefore, we know how to handle the output or output-pattern based on single inputs. It is possible to handle input-patterns by transforming them to features, since an input in the previous architectures can be viewed as a feature. Thus, the only change in the architectures is the transformation of raw inputs to features. As the structure of the experiment is such that every some number of steps the association sets change randomly, this means that the network needs to be capable of learning every possible association set that involves n binary inputs and m binary outputs. Table 1 shows the number of possible association sets for every type of association problem.

Consider a simple case of a many-to-many association problem where $n = 2$ and $m = 1$. Examples of such problems are functions such as OR, AND, as well as the famous XOR problem that was used to signify the importance of hidden units and the use of the backpropagation algorithm (Rumelhart et al., 1986b). The number of possible association sets is $(2^1)^{2^2} = 2^4 = 16$. This means that the requirement enforced on the NN architecture is to be able to learn a randomly selected problem out of these 16 in at most $2^n \times (2^m - 1) = 4$ steps. In supervised learning settings, this would be equivalent to using just one "epoch" to learn the problem. This is a very strict constraint and to the best of our knowledge, the only way an architecture could achieve such a performance is by using a separate feature for every permutation of the

binary inputs. That is, for n binary inputs, the number of features should be 2^n.

Following the work of Rvachev (2013), we model these permutation-detecting feature neurons as follows. Assuming that all inputs are binary, i.e., either 1 (active) or 0 (inactive), and the weights of the connections from inputs to features are either +1 (excitatory) or -1 (inhibitory), a feature neuron that detects a certain permutation computes its activation using the following integration function:

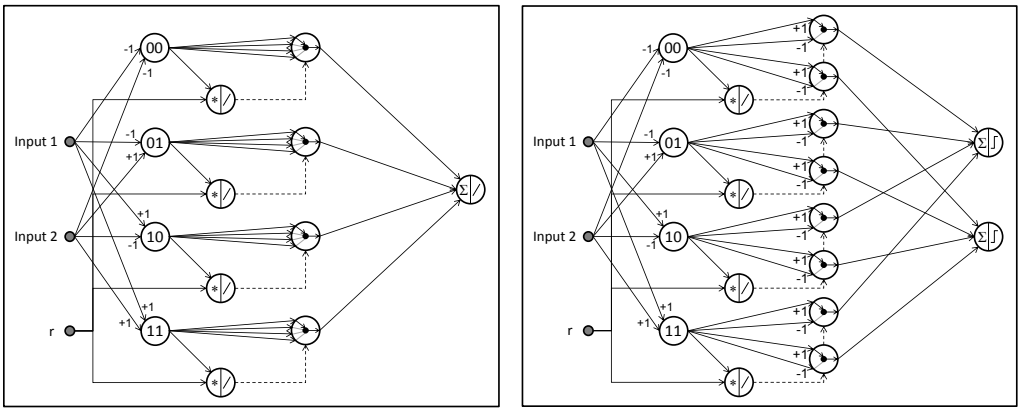
$$a_i^{(std)}(t) = n_i - n_i^* - \overline{n}_i \tag{12}$$

where $n_i^* = \sum_{w_{ji}>0} w_{ji} \geq 0$ is the number of excitatory (standard) connections, $n_i = \sum_{w_{ji}>0} w_{ji} y_j \geq 0$ is the number of active excitatory (standard) connections, $n_i^* \geq n_i \geq 0$, $\overline{n}_i = \sum_{w_{ji}<0} w_{ji} y_j \leq 0$ is the number of active inhibitory (standard) connections. The output of the feature neuron is calculated by feeding its activation through the Heaviside step function:

$$y_i^{(std)}(t) = H(a_i^{(std)}(t)) = \begin{cases} 1 & \text{if } a_i^{(std)}(t) \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

Which permutation of the inputs is detected is dependent on the weights of the incoming connections. More specifically, a weight of +1 from an Input_i results in the detection of a permutation of the input vector where "1" is present at the i_{th} position. Similarly, a weight of -1 from an Input_j results in the detection of a permutation where "0" is present at the j_{th} position. An example where the number of inputs $n = 2$ is shown in Figure 7. In this example, a feature neuron whose connections both have a weight of -1, fires when both inputs are 0; in other words, it detects the permutation "00". If the connection from Input_1 has a weight of -1, but the connection from Input_2 has a weight of +1, the feature neuron fires when the input vector is "01" (i.e., only Input_2 is active). The pattern "10" is detected when the weight of the connection from Input_1 is +1, but from Input_2 is -1, and the pattern "11" is detected when the weights of both connections are +1.

The architectures presented in Figure 7 follow the same logic as the architectures designed for one-to-many association problems (Figure 6). More specifically, the single-output NN of Figure 7a encodes 2^m actions through 2^m connections on each switch neuron, whereas the NN with m output neurons of Figure 7b encodes 2^m actions in the output pattern. Many-to-one association problems can be learned using the architecture of Figure 7a, with



(a) Architecture with one output neuron. (b) Architecture with m output neurons.

Figure 7: Architectures that solve many-to-many association problems for $n = 2$ and $m = 2$. Each of the 2^n input patterns is detected by an individual feature neuron. In order to solve many-to-one association problems, the architecture in (a) can be used, but with m connections on each switch neuron, instead of 2^m.

the difference being that each switch neuron will have m distinct connections instead of 2^m, each one corresponding to an individual action.

3.1.3. Results

Figure 8 illustrates a comparison between the four types of association problems, for $n = 6$ inputs and $m = 6$ outputs. The results show that whenever the association set changes, the corresponding network manages to learn it in an optimal expected number of steps which is: $n \times (m - 1) = 30$ for one-to-one association problems, $n \times (2^m - 1) = 378$ for one-to-many association problems, $2^n \times (m - 1) = 320$ for many-to-one association problems, and $2^n \times (2^m - 1) = 4032$ for many-to-many association problems.

Upon examination of individual trials, we noticed that the networks could learn the associations in fewer than the expected number of steps. This is due to two reasons: (i) the network behavior is deterministic, and (ii) the shuffling of association sets is performed in a random manner resulting in some of them being easier to learn.

3.2. T-maze domains

In this section we investigate tasks in (discrete) T-maze domains that require multiple steps per episode and delayed reward. The name of the maze stems from the fact that it is shaped like the letter "T": an agent is

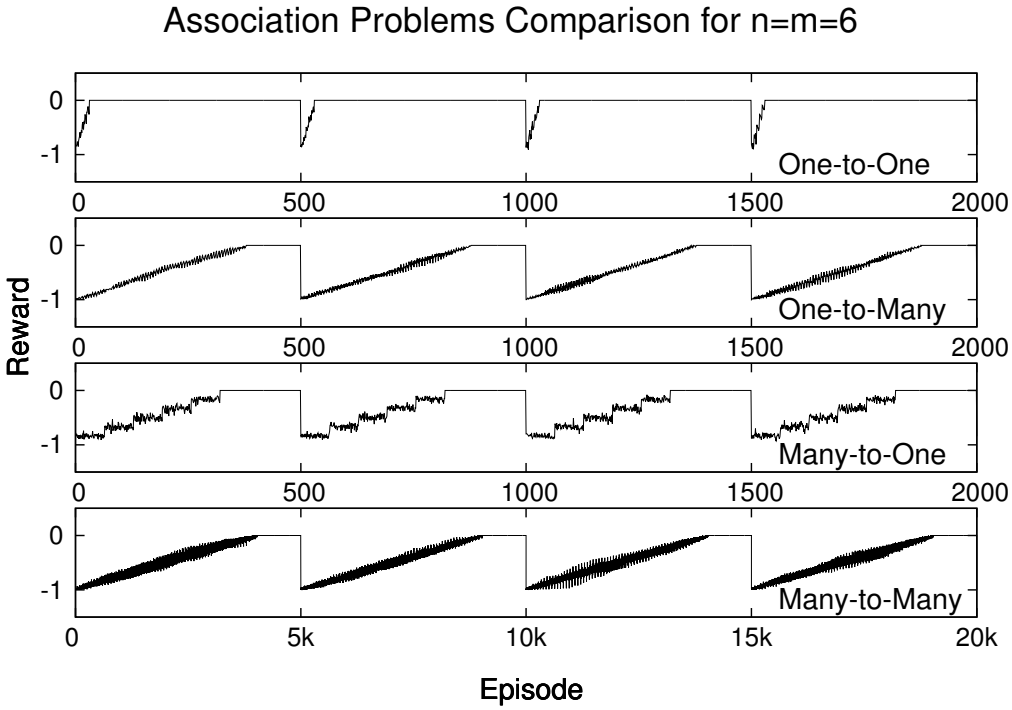


Figure 8: Comparison results for different types of association problems with n=6 inputs and m=6 outputs. The expected number of time steps each NN needs to solve the corresponding problem is: $n \times (m-1) = 30$ for one-to-one association problems, $n \times (2^m - 1) = 378$ for one-to-many association problems, $2^n \times (m-1) = 320$ for many-to-one association problems, and $2^n \times (2^m - 1) = 4032$ for many-to-many association problems. These results are averages over 100 independent trials.

placed at the base of the maze and navigates in a corridor at the end of which there is a turning point that splits the corridor into two branches, one going to the left and the other to the right. Upon reaching a turning point, the agent makes a decision and receives a reward depending on where it ends up. The experiment is performed multiple times and the reward locations can change throughout the experiment. Such environments are often used in animal experiments to assess their memory and learning capabilities.

The basic T-maze can be extended by presenting more turning points in sequence; this can be done by connecting multiple T-mazes. Generally, an n-branched T-maze, i.e., a T-maze with n sequential turning points, has a total of 2^n possible maze-ends. Figure 9 shows an example of a double T-maze environment. The tasks used in experiments with simulated agents

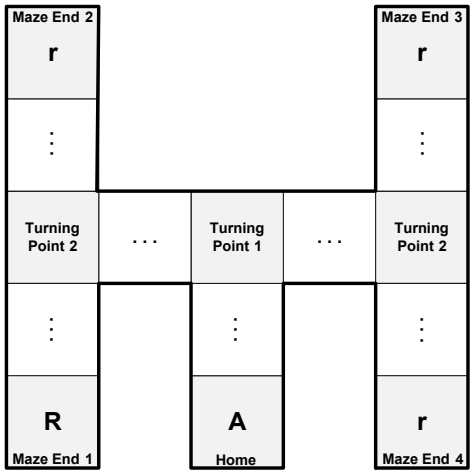


Figure 9: Double T-maze environment. The agent (A) starts at the Home position and needs to navigate towards a Maze End, where a low reward (r) or a high reward (R) will be given. On its way, it will come across two turning points, T1 and T2. There are four possible Maze Ends, and depending on the action taken from each turning point, turning left or right, the agent can explore all of them. The homing task additionally requires from the agent to return to its home position after visiting a Maze End. For example, the agent will end up at Maze End 3, if it takes a right turn at T1 and a left turn at T2, and returns to the home position by taking a right turn at T2 and a left turn at T1.

can be roughly categorized based on (i) whether they use a discrete observation space (e.g., see Soltoggio et al., 2008) or a continuous one (e.g., see Blynel and Floreano, 2003; Risi and Stanley, 2012), and (ii) whether a cue is provided to the agent or not, before reaching some turning point. Experiments with a cue provided to the agent (e.g., see Ulbricht, 1996; Jakobi, 1997; Husbands, 1998; Rylatt and Czarnecki, 2000; Bakker, 2002; Linåker and Jacobsson, 2001a,b; Bergfeldt and Linåker, 2002; Ziemke and Thieme, 2002; Ziemke et al., 2004; Kim, 2004; Rempis, 2007; Littman, 2009; Duarte et al., 2012; Ollion et al., 2012a,b; Lehman and Miikkulainen, 2014; Silva et al., 2014; Duarte et al., 2014) are often used to assess the learning and memory capabilities of an agent or method. Experiments with no cue provided to the agent prior to reaching a turning point (e.g., see Yamauchi and Beer, 1994; Blynel, 2003; Blynel and Floreano, 2003; Gigliotta and Nolfi, 2008; Dürr et al., 2008; Soltoggio, 2008; Soltoggio et al., 2008; Soltoggio and Jones, 2009; Risi et al., 2009, 2010; Risi and Stanley, 2010, 2012; Grouchy and D'Eleuterio, 2014; Lehman and Miikkulainen, 2014; Howard et al., 2014) are often used to additionally assess the agent's exploration capabilities. In

such experiments, the reward locations usually change at some point during the experiment and since the agent has no way of observing that change, the task becomes nonstationary and the agent is required to explore the other reward locations in order to maximize its reward.

3.2.1. Task description

The tasks used in this study are based on experiments performed by Soltoggio et al. (2008) on the evolution of neuromodulated networks, where no cue is provided to the agent. In the *non-homing task*, the agent starts at the bottom of the maze (its home position, H) and is required to navigate to a maze-end (ME), where a reward item is located. At a given time, one of the maze-ends contains a high reward item, while the remaining MEs contain low reward items. In the *homing task*, once the agent reaches a maze-end, it automatically reverses its direction and has the additional requirement of returning back to the home position. We use the standard RL terminology of "episode" to denote a trip from the home position to a ME (and back, in the homing scenario), and "trial" to denote a lifetime of the agent.

An important feature of the task is that the location of the high reward is not kept fixed, but changes to a random location during the lifetime of the agent making the problem nonstationary. Given an n-branched T-maze, an optimal agent would need on average 2^n episodes to explore all 2^n possible MEs to find the one that contains the high reward.

The agent's observation consists of four variables each encoded by an input neuron: (1) H is set to 1.0 when the agent is at the home position, (2) T is set to 1.0 when the agent is at a turning point, (3) ME is set to 1.0 when the agent is at a maze-end, and (4) r is the amount of reward collected at a maze-end. The agent is allowed to perform three actions, i.e., *turn left*, *go forward* or *turn right*, that are encoded by a single output neuron which uses the weighted-sum integration function and the hyperbolic tangent activation function. More specifically, the action "turn left" is selected if the activation of the output neuron $o \leq -0.33$, the action "go forward" is selected if $-0.33 < o \leq 0.33$, and "turn right" is selected if $o > 0.33$. We chose to use this kind of architecture and not, for example, one where the action is encoded by a softmax layer, in order to be consistent with previous work (Soltoggio et al., 2008). No noise affects the inputs or neural transmission.

The reward function used is the same as in Soltoggio et al. (2008): the value of the high reward is 1.0, whereas the low reward is 0.2. During navigation the reward is set to 0. The agent is penalized for crashing on the walls

and this happens when it fails to maintain a forward direction in corridors, or when it fails to turn to the appropriate direction at a turning point. The penalty for crashing is 0.4 and this value is subtracted from the amount of reward collected. When this happens the agent is repositioned at the home location and a new episode commences. In the homing task, there is an additional penalty for failing to return to the home position. This happens when the agent while navigating back to the home position, at a turning point it enters a corridor that is not the one from which it came from. The penalty for doing so is 0.3 and this also signifies a terminal state of the episode and the agent is relocated to the home position. The corridors could stretch for a variable number of agent steps; in our experiments, this number was set to 2 in all corridors. The turning point stretches for only one agent step.

It is worth noting that the environment is partially observable since the agent does not know its exact location in the maze and the high reward location. As it will be shown below, both the partial observability problem and the problem of nonstationarity are solved with the NN architectures we constructed. More specifically, the problem of partial observability is resolved by carefully constructing a neural module that detects certain features in the environment. The problem of nonstationarity is addressed by making the agent explore the different maze-ends using the idea of modulated switch neurons we present in this work.

3.2.2. NN architectures

Let us approach the single T-maze problem first. We exploit the observation that an optimal agent goes forward inside the corridors and only turns when its T input is active. When its ME input becomes active, the high reward does not elicit a change in behavior, whereas the low reward does. A change in behavior only means, in this case, to turn left or right at the turning point, depending on whether it turned right or left respectively at the previous episode. A simple way to implement this behavior is by connecting the T input with a switch neuron using two connections, one for each turning action, and the switch neuron to the output neuron using a weight of 1. We know that the output neuron uses the hyperbolic tangent function and that "turn left" is selected if the output $o \leq -0.33$ and "turn right" is selected if $o > 0.33$. This means that by choosing appropriate weights, a negative weight for "turn left" and a positive one for "turn right", the actions can be executed when the T input becomes active. If the T input is inactive, the action "go forward" is executed since the output values of the switch neu-

ron and the output/action neuron remain 0. We choose these weights to be high in magnitude to saturate the output neuron; any weights, however, that satisfy the above inequalities would work. These are -5 for "turn left" and $+5$ for "turn right", and result in an output of ≈ -1 and $\approx +1$ respectively. What is now missing from the model is how to modulate the switch neuron.

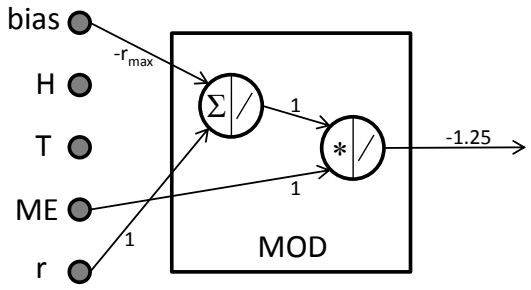


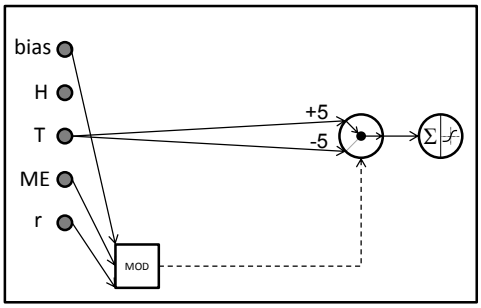
Figure 10: Neural circuit for converting the reward signal to a modulatory one that is compatible with switch neurons. This circuit implements the equation $ME(r-r_{max})$ where $r_{max} = 1$ is the high reward and the maximum obtained in the task. The multiplication with the input ME is done since we want the modulation to be applied only when ME is active, i.e., at the maze-end; thus, ME acts as a gate. The output is either 0.0, when the high reward is obtained, or -0.8, when the low reward (=0.2) is obtained. Multiplying the output of -0.8 with a weight value of -1.25 results in the value of +1. Therefore, when the low reward is received the output of the circuit is +1 and when the high reward is received the output of the circuit is 0.

Figure 10 shows a simple neural circuit that converts the reward signal to one that is able to modulate the switch neurons. This circuit implements the following function

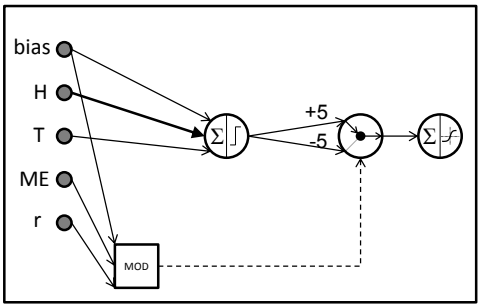
$$y = ME(r - r_{max}) \tag{14}$$

where r is the value of the reward input and $r_{max} = 1$ is the high reward and the maximum obtained in this task. This equation essentially converts all reward signals to a modulation of 0, apart from when the low reward is obtained in which case the modulatory signal becomes equal to +1.

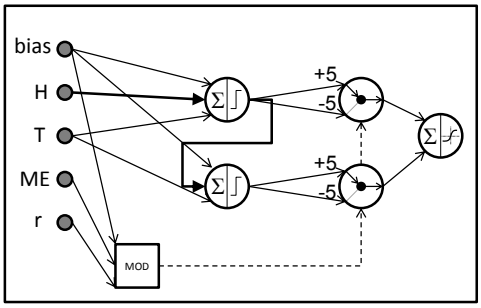
Figure 11a illustrates the simple architecture described above for solving the single (non-homing) T-maze task. In order to create architectures for the more complicated n-branched T-mazes it is required to address the issue of perceptual aliasing that stems from the limited sensing capabilities of the agent. Our approach is based on feature detector neurons. In particular, we use a separate feature detector for each sequential turning point. Figure 11b shows an alternative architecture that solves the single T-maze task using



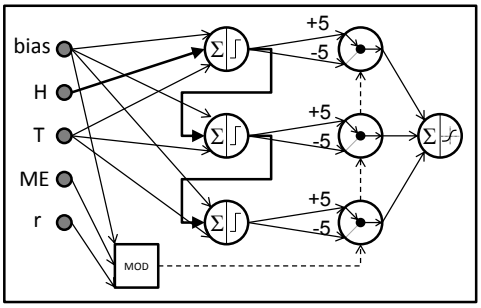
(a) Single T-maze architecture without a feature detector neuron.



(b) Single T-maze architecture with a feature detector neuron.



(c) Double T-maze architecture.



(d) Triple T-maze architecture.

Figure 11: Architectures for the single, double and triple T-maze (non-homing) task. *Solid* lines represent standard connections, *dashed* lines represent modulatory connections and a *thick solid* line towards a neuron i represents a vector of delayed standard connections $[w_{1i}(t-1), w_{2i}(t-2), ..., w_{\delta i}(t-\delta)]^T$ where δ = maximum corridor length + 1. The weights of the connections of the modulatory circuit are shown in Figure 10. The weights of the connections from the bias unit to the feature detectors are all -1.5. All other weights are 1.0. The figures show how straightforward it is to extend the neural architecture as the number of sequential decision points increases.

a hidden neuron that fires when the agent comes from the home position and reaches the turning point. The feature detector uses the weighted-sum integration function and the Heaviside activation function. The figure shows that there are incoming connections from three different inputs: the bias unit that has a constant value of +1, the turning point input, and the home input. The weight of the connection from the bias unit is -1.5. This inhibition is used to suppress the activity of the neuron and by using a weight of +1 on the remaining connections, the feature neuron fires only when a signal of +1 comes from both H and T, since the activation becomes $0.5 > 0.0$. In

order to achieve this, the connection from the H input needs to have a delay. Since we use the same time scale for both the agent-environment interaction and the network activation, the delay of the connection must be equal to the corridor length + 1. We followed a more robust approach where the connection from the H input is not a single delayed connection, but a vector of delayed connections $[w_{1i}(t-1), w_{2i}(t-2), ..., w_{\delta i}(t-\delta)]^T$ where i is the index of the neuron and δ = maximum corridor length + 1. This approach implements what is known in the literature as a *complete serial-compound* stimulus (Sutton and Barto, 1990; Montague et al., 1996; Schultz et al., 1997; Gershman et al., 2013) and comprises a way of representing a stimulus through time. Its advantage is that it can be used for mazes with a variable corridor length; we only need to know the maximum corridor length[1].

Figures 11c and 11d illustrate how the architectures can be extended to solve the double and triple T-maze tasks respectively. For the double T-maze task, a new feature detector neuron needs to be added that detects the new (second) turning point in the sequence. This feature detector neuron needs to fire when the agent comes from the previous (first) turning point and reaches the new (second) one. This means that instead of being connected to the H input with a vector of delayed connections, it needs to connect to the previous (first) feature detector, as the latter fires when the previous (first) turning point is encountered. The new feature neuron needs to connect to its own switch neuron and the switch neuron to the output neuron. The trick now is not to modulate both neurons in parallel (as in Figure 4a), but in sequence (as in Figure 4b). This is because we want the agent to be able to explore all maze-ends one after the other. For the triple T-maze and generally an n-branched T-maze, the procedure above is repeated to create an architecture that can optimally solve the corresponding problem.

For the homing task we make the following observation. The agent takes a series of decisions when going towards a maze-end (the first part of the task) and reverses them when going towards the home position (the second part of the task). The architectures for the non-homing tasks are simple to interpret, since we know the exact function of each neuron and the role each plays on the synthesis of the behavior of the agent. Since the homing task has

[1]Note that the feature detection part could be done by using recurrent connections, obviating the requirement of knowing the maximum corridor length. However, feature detection is not the focus of this work and the approach described above suffices.

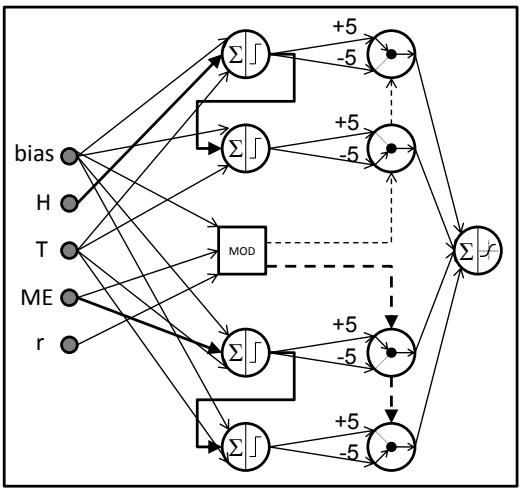


Figure 12: An architecture for the double T-maze with homing task. When going towards a maze-end, this architecture makes the agent turn right at T1 (by selecting the connection with weight +5) and left at T2 (by selecting the connection with weight -5), thus, visiting ME3. When homing, the agent turns right at T2 and left at T1.

a second part, which seems to be mirroring the first, this mirroring should appear in an architecture that solves the homing task. That is, the NN architecture should have some symmetry that reflects the symmetry of the task. This turns out to be true, as shown in Figure 12, where a NN that solves the double T-maze homing task is illustrated. The lower subnetwork of this architecture encodes the behavior of the agent for the reverse direction (from a maze-end to the home position). For this subnetwork, it is worth noting that: (i) its upper feature detector neuron is connected to the ME input, as the agent now starts from a maze-end, (ii) its modulatory connections are delayed (shown with a thicker line), so that the change to the switch happens *after* the agent leaves the corresponding turning point, and (iii) the states of the switch neurons for the corresponding turning point are reversed, in order to reverse the "turn" actions accordingly.

It is important to note that the states of all neurons apart from the switch and integrating neurons are reset at the beginning of each episode. This is done because we want to model an *adaptive* solution to the problem, and not a solution that depends on memory (from recurrent/delayed connections) between episodes. Thus, the delayed connections in our NNs are only used *during* episodes (not between episodes). The switch neurons are not reset because their role is similar to the role of a synaptic plasticity rule. What

we mean with this is that when a synaptic plasticity rule modifies a weight, that weight is not reset to its initial value at the beginning of each episode. Equivalently, when a switch neuron selects a different weight, it does not reset its selection at the beginning of a new episode. The integrating neurons are not reset, in order to be able to modulate switch neurons that come next on the modulatory pathway and therefore, for exploration to work correctly.

3.2.3. Results for the double T-maze tasks

The behavior of the agent in the double T-maze environment is shown in Figure 13. The same behavior can be observed for both the non-homing and the homing tasks. The gray shaded area represents the location of the high reward and the black dots indicate the maze-end explored by the agent at the corresponding episode. The high reward location changes every 20 episodes using the schedule $(1, 2, 3, 4)$, meaning that it is at ME1 in episodes 1-20, at ME2 in episodes 21-40, at ME3 in episodes 41-60, and at ME4 in episodes 61-80. This schedule was given on purpose, since it is the only one that makes the agent explore all three suboptimal maze-ends every time the high reward location changes, before finding the optimal one. That is because of the initial states of the switch neurons and the way the architecture is designed, which endow the agent with the following cyclic exploration pattern: $(4, 3, 2, 1)$. This means that the agent first explores ME4. If the high reward is not located there, then during the next episodes, it will explore ME3, then ME2, and ME1. If the high reward is not at ME1 either, the agent starts again at ME4 and follows the above exploration procedure. This is shown in Figure 13, where the agent settles at ME1 in episode 4 after exploring ME4, ME3 and ME2 in this order. In episode 21 the agent visits ME1 again, since it has no way of knowing that the high reward location has changed. It then resets its exploration pattern by visiting ME4 in episode 22, then ME3 in episode 23, and finally settles at ME2 from episode 24 until episode 41 at which point it needs to explore again. Note that if the reward location changes randomly, the agent could find it in less than four episodes. In any case, whenever the agent finds the reward location, it continues to go there.

4. Discussion

In summary, a switch neuron can be seen as a regulatory gate that allows information to pass through only from a single incoming connection. Modulatory signals change the modulatory activation of the switch neuron by a

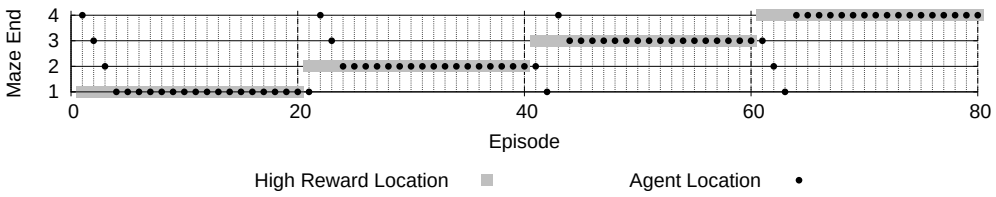


Figure 13: Agent behavior in the double T-maze environment. The same behavior can be observed for both the non-homing and the homing tasks. The gray shaded area represents the location of the high reward and the black dots indicate the maze-end explored by the agent at the corresponding episode. The high reward location changes every 20 episodes using the schedule $(1, 2, 3, 4)$. This schedule was given on purpose, since it is the only one that makes the agent explore all three suboptimal maze-ends every time the high reward location changes, before finding the optimal one.

certain amount. If this amount is enough, it forces the switch to change connection. If we imagine the switch like a wheel, enough positive modulatory activation will push the switch in a clockwise manner, thus, selecting the next connection, while enough negative modulatory activation will pull the switch in the opposite, counterclockwise direction, effectively selecting the previous connection. Therefore, it is the resulting modulatory activity that encodes the selected connection and not the modulatory signals themselves, as the latter encode the change in the modulatory activity. This seems to be useful in situations where a target behavior cannot be directly decided as a function of the input, and some exploration is needed to discover it.

In addition, we showed that a separate pathway we call "modulatory pathway" can be used to link circuits of switch neurons in a sequence of gating events, with each circuit appropriately modulating the next. This confirms a theoretical model of Gisiger and Boukadoum (2011), where they hypothesized that such a pathway, formed by the gating mechanisms, could convey other types of information and might be responsible for the production of structured behavior. Interestingly, as the results in this work clearly show, when this pathway is embedded in appropriate architectures, it has the property of implementing optimal, deterministic exploration in binary association tasks and discrete T-maze problems. Therefore, the proposed switch neuron computational model can be used to generate such optimal adaptation behaviors and it would be interesting to see how it performs in other tasks. The modulatory pathway is just an example of a modulatory topology and worked well for the tasks presented in this work. More complex tasks

might require a different modulatory topology amenable to optimization.

The modulatory pathway implies that the switch neurons modulate each other in a sequence, and this is implemented by replacing each switch neuron with the switch module. It is reasonable to ask: is this 'switch module' needed? Certainly they are needed, as firstly it can easily be shown (not presented) that the architectures that use the switch modules are more efficient than the ones that only use switch neurons in terms of number of connections in one-to-many or many-to-many association tasks. Secondly, these modules seem to be essential in sequential decision making tasks, as illustrated by the architectures for the T-maze problems (see Figures 11 and 12).

Although the architectures were manually designed, they nevertheless provide evidence that such an approach is effective. It is often hard to design a good architecture that works well for varying sets of problems. Therefore, it surely is beneficial to learn these architectures, instead of designing them by hand. A question then is how can such architectures be learned or automatically designed? A possible solution comes from evolutionary computation. The field of neuroevolution (i.e., evolutionary NNs) has progressed in recent years, especially in the area of generative and developmental systems, where the genotype does not map directly to the phenotype but goes through a process of development. For example, an algorithm called Hypercube-based NeuroEvolution of Augmenting Topologies (HyperNEAT, Stanley et al., 2009), can evolve types of networks called compositional pattern producing networks (CPPNs, Stanley, 2007) that usually contain few connections, but are capable of generating regular (Clune et al., 2011) and modular NNs (Verbancsics and Stanley, 2011; Huizinga et al., 2014) with many more connections. This is because CPPNs can encode connectivity patterns that contain various types of regularities, such as symmetry, imperfect symmetry, repetition and repetition with variation. The architectures we designed in this study display regularities that seem to be easily found by algorithms such as HyperNEAT. Therefore, an interesting future work would be to use such algorithms to evolve switch neuron architectures. As the modulatory connectivity seems to be linked to optimal exploration behavior in the investigated tasks, it would be particularly interesting to see what modulatory topologies emerge in other tasks and how adaptive the final solutions are, compared to recurrent or plastic NNs without switch neurons.

Related to the above, it is worth noting that the proposed switch neuron model does not come to replace existing mechanisms that are already successful in T-maze tasks (such as the neuromodulation approach of Soltoggio et al.,

2008) and association problems (see Soltoggio and Stanley, 2012; Tonelli and Mouret, 2013), but rather to complement them. Comparing the existing approaches with the switch neuron model could possibly reveal that the switch neuron model provides higher evolvability, especially if the evolutionary algorithm uses an indirect encoding (Yao, 1999) that promotes regular and/or modular structures. This demands a thorough empirical investigation which will establish the advantages and disadvantages of switch neurons and existing mechanisms. Note that care needs to be taken when attempting to evolve adaptive behavior. This is because it is a problem that presents a number of deceptive traps which could be mitigated by appropriately designing the fitness function (Soltoggio and Jones, 2009) or changing the way the evolutionary algorithm works by searching for novel behaviors (Lehman and Stanley, 2008, 2011; Lehman and Miikkulainen, 2014; Risi et al., 2009, 2010) or encouraging behavioral diversity (Mouret and Doncieux, 2012).

An interesting research direction is the development of machine learning methodologies or learning rules that optimize switch neuron networks based on given datasets. For example, a gradient-descent-type of algorithm could potentially be used to fit a switch neuron network on a dataset that contains nonstationary sequential data, i.e., where the data records at certain points are generated from different distributions. Such networks could provide better generalization performance, as they could explore various neuronal combinations online, even in a testing/recognition phase. This, however, requires modulatory signals to be available. These signals could potentially be a function of the inputs. For example, some feature modules with modulatory output connections could change the state of switch neurons and route information flow differently. These signals could also be calculated by neural circuits that take into account some: (i) input reconstruction error, calculated using an "autoencoder" (Rumelhart et al., 1986a; Bourlard and Kamp, 1988) / "replicator" (Hecht-Nielsen, 1995) subnetwork; (ii) feature prediction error, calculated using a "forward model" (Schmidhuber, 1991; Miall and Wolpert, 1996) or "general value function" (Sutton et al., 2011) subnetwork; (iii) RPE, calculated using a "critic" (Sutton and Barto, 1998) subnetwork; (iv) error calculated through gradient descent during the recognition phase, for finding not the weights but the neural activations (Achler, 2014). The learning rules could potentially be found using evolutionary algorithms, as in our previous work (Vassiliades and Christodoulou, 2013).

Regarding the association tasks, if viewed from a supervised learning perspective, many-to-one tasks can be seen as multi-class classification problems,

while many-to-many tasks can be seen as multi-label classification problems. In those settings, however, there is a target output and consequently an error, both of which are vectors with dimension equal to the dimension of the output vector. In the settings presented in this work, this error is not a vector, but just a scalar signal, i.e., the modulatory signal. This is the reason why these tasks are not solved in just a single time step and require multiple time steps. Our architectures solve the problems using the optimal number of steps, at the expense of an exponential (2^n, where n is the number of inputs) number of features. This is because the tasks presented in this work were not designed to have any relationships between the inputs, as they are allowed to change abruptly. For instance, consider a parity problem with n inputs. It is possible to solve it with a learning algorithm and an architecture that uses less than 2^n features, at the expense of some time. The parity problem, however, is just a single function, and constructing a network that is able to solve all possible functions of n binary inputs in an optimal number of steps requires an increase to the capacity of the network.

Related to the above, suppose that the inputs are real-valued instead of binary, but still independent. This means that we could discretize them at some target resolution, treat each discretization as a separate "input" and devise a feature layer in which the neurons detect permutations of these discretized inputs. This would still solve the problem in an optimal expected number of steps. Real-world problems, however, do not change as abruptly as the ones presented here, and there are relationships between the inputs, so there is no need for such expanded feature layers. For this reason, the inputs in our architectures should not be viewed as raw inputs, but rather as features deep in the network. Thus, switch neuron architectures could complement approaches that learn feature representations by stacking the switch neurons or switch modules at the very last layers.

While this might seem promising, it is also worth considering the use of switch neurons (and layers of switch neurons) between feature layers and not just at the last layers during learning. This is because switch neurons can be considered as specialized gating mechanisms, and this is not the first time gating mechanisms have been used in NNs. In fact, examples can be found in many works in the literature: higher-order NNs (Rumelhart et al., 1986a; Giles and Maxwell, 1987; Giles et al., 1988; Durbin and Rumelhart, 1989; Shin and Ghosh, 1991; Leerink et al., 1995) that use product units; long short-term memory cells (Hochreiter and Schmidhuber, 1997; Gers et al., 2000) that were created to address the vanishing or exploding gradient problem

when training recurrent NNs; committee machines with a dynamic structure, such as mixture and hierarchical mixture of experts (Jacobs et al., 1991; Jordan and Jacobs, 1994); binary stochastic units used in Boltzmann machines (Ackley et al., 1985) that can implement conditional computation (Bengio et al., 2013); an approach called dropout that samples an ensemble of NNs from a single NN by stochastically gating (deactivating) hidden units during training (Srivastava et al., 2014); and more recent works (Bengio, 2013; Cho et al., 2014; Chung et al., 2014, 2015; Srivastava et al., 2015).

Moreover, we can see a relationship between gating and time scale, as pathways can be selectively activated at specific time steps, while others at steps that have a slower or faster clock rate (Koutník et al., 2014). The idea behind this approach is related to hierarchical learning and goes back to Ashby (1952) who proposed a gating mechanism to handle repetitive situations. In Ashby's work, it is assumed that an agent accumulates adaptations in the form of new behaviors which can be switched depending on some "variables" that operate "at a much slower order of speed" (Ashby, 1952). This is also related to multi-task learning since different environmental conditions might require adaptations and the accumulation of new behaviors. This is shown in an abstracted form in our experiments with the association tasks, where the designed architectures were used to learn families of tasks and not just a single task (e.g., learning not just the XOR function, but all functions that involve 2 binary inputs and 1 binary output). Switch neurons naturally encourage modularity and this is evident from all the architectures we presented. Neural modularity was found to encourage the learning of new skills without forgetting old skills (Ellefsen et al., 2015), a property that is crucial for hierarchical and multi-task learning. Therefore, switch neurons could potentially be used in both hierarchical and multi-task learning situations.

It is worth mentioning an approach presented by Rvachev (2013) which bears some similarities to our work. In that paper, it was proposed that a neuron model could classify input permutations. This was achieved by building upon a previous compartmental model of a hippocampal pyramidal neuron (Poirazi et al., 2003). More specifically, Rvachev (2013) modeled synaptic clusters that form on dendritic branches via projections from input neurons. A cluster is excited if input neurons of a particular permutation fire. If this local excitation is followed by the post-synaptic neuron firing, a back-propagating action potential (bAP) at that site and, subsequently, a positive RPE-type signal, then the expression of a "combinatorial memory" component on the cluster is reinforced, otherwise, it is weakened if a negative

RPE-type signal is incurred. To make learning possible, a training phase is performed, where a "guessing input" on the post-synaptic neuron is used to make the neuron fire, while it is assumed that this firing always causes a bAP. This approach has only been used to solve certain stationary many-to-one association problems. In Section 3.1.2 we utilized a similar permutation-detecting mechanism as part of the activation function of feature detector neurons (rather than synaptic clusters). As in our model, the model by Rvachev (2013) is much more efficient in training time, since only one "epoch" is needed for classifying input patterns, but much less efficient in the number of connections used compared to other methods.

Finally, we would like to discuss a potential relationship between switch neuron architectures and synaptic plasticity rules. One-to-many association tasks have been used in the work of Soltoggio and Stanley (2012) in which a network architecture and a synaptic plasticity rule, called reconfigure-and-saturate (R&S) Hebbian plasticity, were introduced. The R&S rule, which relies on noise to explore weight configurations, was found to learn one-to-many associations tasks with $n = m = 6$ in an optimal number of steps (see Soltoggio and Stanley, 2012, Figure 14B), thus, comparable to our results (see Figure 8). We experimented with the source code provided for that work[2] and confirmed the results. However, we also found that for a larger number of outputs (e.g., 10) the rule does not learn the associations in the optimal number of steps as the switch neuron architectures do. This is because its exploration mechanism is stochastic, whereas the exploration mechanism of the switch neuron architectures is deterministic. Although the R&S rule was not designed with this guarantee in mind, its strength lies in its ability to work under stochastic modulation policies. In contrast, the switch neuron architectures of this work rely on precise modulation signals in order to function correctly, as it is assumed that noise is filtered before entering the modulatory subnetwork. Thus, we could say that switch neuron architectures, while being artificial, they are nature-inspired in the sense that they try to model the effect of a rule such as the R&S rule, and consequently the phenomenon of behavioral plasticity, without modeling the dynamics of synaptic plasticity. Since the behaviors are essentially hard-wired in the architectures we presented, future work could explore whether the synergy between switch

[2]The source code was downloaded from: http://andrea.soltoggio.net/rec-sat [last accessed: 18 June 2015].

neurons and neural plasticity mechanisms (such as structural and synaptic plasticity) could discover behaviors that are not initially encoded in the network. In addition, an interesting future direction would be the design of novel switch neuron architectures or the extension of the basic switch neuron model in order to handle noisy signals.

5. Conclusion

This paper introduced a computational model of an artificial neuron, called switch neuron, that endows an agent with the ability of switching between different behaviors in response to environmental changes. The model works by selectively gating its incoming connections in a way that permits the flow of information from only one of them. This connection is determined by the level of modulatory activation of the neuron which is affected by modulatory signals. While these signals could encode some information about the reward, they are qualitatively different in the sense that both positive and negative signals are interpreted as 'instructions' to change the selected connection, and the sign determines the direction of change. An important design aspect of the switch neuron is that it can modulate other switch neurons. This is done by using the switch neuron as part of a three-neuron module with the others being: (i) a 'modulating' neuron, that collects the incoming modulatory signals and modulates the switch neuron, and (ii) an 'integrating' neuron, that integrates the modulatory signal emitted to the switch neuron, fires above a threshold and is used to 'communicate' with other switch modules or individual switch neurons. We showed that a topology where these switch modules are placed sequentially on the same modulatory pathway explores in a principled manner all permutations of the connections arriving on the switch neurons.

The model was tested in two sets of tasks, namely nonstationary association tasks and T-maze domains. We presented appropriate switch neuron architectures that can learn one-to-one, one-to-many, many-to-one and many-to-many binary association tasks, and discussed how to extend them for an arbitrary number of inputs and outputs. Architectures were presented for the homing and non-homing T-maze tasks as well, where we discussed how to extend them for an arbitrary number of sequential decision points. For all tasks, the architectures were clearly shown to generate optimal adaptive behaviors, thus, providing evidence that the switch neuron model could be a valuable tool in simulations where behavioral plasticity is required.

Acknowledgments

We would like to thank Andrea Soltoggio for discussions on neural plasticity and T-maze domains, and the three anonymous reviewers for their constructive comments.

References

Achler, T., 2014. Symbolic neural networks for cognitive capacities. Biologically Inspired Cognitive Architectures, 9, 71–81.

Ackley, D. H., Hinton, G. E., Sejnowski, T. J., 1985. A learning algorithm for Boltzmann machines. Cognitive Science, 9 (1), 147–169.

Anderson, C. H., Van Essen, D. C., 1987. Shifter circuits: a computational strategy for dynamic aspects of visual processing. Proceedings of the National Academy of Sciences, 84 (17), 6297–6301.

Arnold, S. F., 2011. Neuro-cognitive organization as a side-effect of the evolution of learning ability. In: Proceedings of the 2011 IEEE Symposium on Artificial Life (ALIFE 2011). IEEE, Piscataway, NJ, pp. 100–107.

Arnold, S. F., Suzuki, R., Arita, T., 2012. Second order learning and the evolution of mental representation. In: Adami, C., Bryson, D. M., Ofria, C., Pennock, R. T. (Eds.), Artificial Life 13: Proceedings of the Thirteenth International Conference on the Simulation and Synthesis of Living Systems. MIT Press, Cambridge, MA, pp. 301–308.

Arnold, S. F., Suzuki, R., Arita, T., 2013. Evolution of social representation in neural networks. In: Liò, P., Miglino, O., Nicosia, G., Nolfi, S., Pavone, M. (Eds.), Advances in Artificial Life, ECAL 2013: Proceedings of the Twelfth European Conference on the Synthesis and Simulation of Living Systems. MIT Press, Cambridge, MA, pp. 425–430.

Ashby, W. R., 1952. Design for a Brain: The Origin of Adaptive Behavior. Chapman and Hall, London, UK.

Bakker, B., 2002. Reinforcement Learning with Long Short-Term Memory. In: Dietterich, T. G., Becker, S., Ghahramani, Z. (Eds.), Advances in Neural Information Processing Systems 14 (NIPS 2001). MIT Press, Cambridge, MA, pp. 1475–1482.

Barbas, H., Zikopoulos, B., 2007. The prefrontal cortex and flexible behavior. Neuroscientist, 13 (5), 532–545.

Barto, A., 1995. Adaptive critics and the basal ganglia. In: Hook, J. C., Davis, J. L., Beiser, D. G. (Eds.), Models of Information Processing in the Basal Ganglia. MIT Press, Cambridge, MA, Ch. 11, pp. 215–232.

Bengio, Y., 2013. Deep learning of representations: Looking forward. In: Dediu, A. H., Martín-Vide, C., Mitkov, R., Truthe, B. (Eds.), Statistical Language and Speech Processing. Vol. 7978 of Lecture Notes in Computer Science. Springer, Berlin, pp. 1–37.

Bengio, Y., Léonard, N., Courville, A. C., 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. CoRR, abs/1308.3432. [online at http://arxiv.org/abs/1308.3432, accessed 12 June 2015]

Bergfeldt, N., Linåker, F., 2002. Self-organized modulation of a neural robot controller. In: Proceedings of the 2002 International Joint Conference on Neural Networks (IJCNN 2002). IEEE, Piscataway, NJ, pp. 495–500.

Bi, G.-Q., Poo, M.-M., 1998. Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type. Journal of Neuroscience, 18 (24), 10464–10472.

Binder, M. D., Hirokawa, N., Windhorst, U., 2009. Encyclopedia of Neuroscience. Springer, Berlin.

Bliss, T. V., Lømo, T., 1973. Long-lasting potentiation of synaptic transmission in the dentate area of the anaesthetized rabbit following stimulation of the perforant path. Journal of Physiology, 232 (2), 331–356.

Blynel, J., 2003. Evolving reinforcement learning-like abilities for robots. In: Tyrrell, A. M., Haddow, P. C., Torresen, J. (Eds.), Evolvable Systems: From Biology to Hardware. Vol. 2606 of Lecture Notes in Computer Science. Springer, Berlin, pp. 320–331.

Blynel, J., Floreano, D., 2003. Exploring the T-maze: Evolving learning-like robot behaviors using CTRNNs. In: Cagnoni, S., Johnson, C. G., Cardalda, J. J. R., Marchiori, E., Corne, D. W., Meyer, J.-A., Gottlieb, J., Middendorf, M., Guillot, A., Raidl, G. R., Hart, E. (Eds.), Applications of

Evolutionary Computing. Vol. 2611 of Lecture Notes in Computer Science. Springer, Berlin, pp. 593–604.

Bourlard, H., Kamp, Y., 1988. Auto-association by multilayer perceptrons and singular value decomposition. Biological Cybernetics, 59 (4-5), 291–294.

Burchell, T. R., Faulkner, H. J., Whittington, M. A., 1998. Gamma frequency oscillations gate temporally coded afferent inputs in the rat hippocampal slice. Neuroscience Letters, 255 (3), 151–154.

Buxton, D., Bracci, E., Overton, P. G., Gurney, K., 2015. Substance P release in the striatum allows for efficient switching between distinct actions in action sequences. In: Proceedings of the Integrated Systems Neuroscience Workshop. Manchester, UK, p. 33.

Cho, K., van Merrienboer, B., Gülçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y., 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In: Moschitti, A., Pang, B., Daelemans, W. (Eds.), Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014). ACL, Stroudsburg, PA, pp. 1724–1734.

Chung, J., Gülçehre, Ç., Cho, K., Bengio, Y., 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. CoRR, abs/1412.3555. [online at http://arxiv.org/abs/1412.3555, accessed 12 June 2015]

Chung, J., Gülçehre, Ç., Cho, K., Bengio, Y., 2015. Gated feedback recurrent neural networks. CoRR, abs/1502.02367. [online at http://arxiv.org/abs/1502.02367, accessed 12 June 2015]

Churchland, P. S., Sejnowski, T. J., 1992. The Computational Brain. MIT press, Cambridge, MA.

Cisek, P., Puskas, G. A., El-Murr, S., 2009. Decisions in changing conditions: the urgency-gating model. Journal of Neuroscience, 29 (37), 11560–11571.

Clune, J., Stanley, K. O., Pennock, R. T., Ofria, C., 2011. On the performance of indirect encoding across the continuum of regularity. IEEE Transactions on Evolutionary Computation, 15 (3), 346–367.

Coleman, O. J., Blair, A. D., Clune, J., 2014. Automated generation of environments to test the general learning capabilities of ai agents. In: Proceedings of the 2014 Conference on Genetic and Evolutionary Computation. GECCO '14. ACM, New York, NY, pp. 161–168.

Duarte, M., Oliveira, S., Christensen, A. L., 2012. Hierarchical evolution of robotic controllers for complex tasks. In: Proceedings of the 2012 IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL 2012). IEEE, Piscataway, NJ, pp. 75–80.

Duarte, M., Oliveira, S. M., Christensen, A. L., 2014. Evolution of hybrid robotic controllers for complex tasks. Journal of Intelligent & Robotic Systems, doi: 10.1007/s10846–014–0086–x.

Durbin, R., Rumelhart, D. E., 1989. Product units: A computationally powerful and biologically plausible extension to backpropagation networks. Neural Computation, 1 (1), 133–142.

Dürr, P., Mattiussi, C., Floreano, D., 2010. Genetic representation and evolvability of modular neural controllers. IEEE Computational Intelligence Magazine, 5 (3), 10–19.

Dürr, P., Mattiussi, C., Soltoggio, A., Floreano, D., 2008. Evolvability of neuromodulated learning for robots. In: Stoica, A., Tunstel, E., Huntsberger, T., Arslan, T., Vijayakumar, S., El-Rayis, A. O. (Eds.), Proceedings of the ECSIS Symposium on Learning and Adaptive Behaviors for Robotic Systems (LAB-RS 2008). IEEE Computer Society Press, Los Alamitos, CA, pp. 41–46.

Ellefsen, K. O., 2013. Evolved sensitive periods in learning. In: Liò, P., Miglino, O., Nicosia, G., Nolfi, S., Pavone, M. (Eds.), Advances in Artificial Life, ECAL 2013: Proceedings of the Twelfth European Conference on the Synthesis and Simulation of Living Systems. MIT Press, Cambridge, MA, pp. 409–416.

Ellefsen, K. O., Mouret, J.-B., Clune, J., 2015. Neural modularity helps organisms evolve to learn new skills without forgetting old skills. PLoS Computational Biology, 11 (4), e1004128, doi: 10.1371/journal.pcbi.1004128.

Floresco, S. B., Grace, A. A., 2003. Gating of hippocampal-evoked activity in prefrontal cortical neurons by inputs from the mediodorsal thalamus and ventral tegmental area. Journal of Neuroscience, 23 (9), 3930–3943.

Gerfen, C., Wilson, C., 1996. The basal ganglia. In: Swanson, L. W., Björklund, A., Hökfelt, T. (Eds.), Integrated systems of the CNS, Part III. Vol. 12 of Handbook of Chemical Neuroanatomy. Elsevier, Amsterdam, The Netherlands, Ch. 2, pp. 371–468.

Gers, F. A., Schmidhuber, J., Cummins, F., 2000. Learning to forget: Continual prediction with LSTM. Neural Computation, 12 (10), 2451–2471.

Gershman, S. J., Moustafa, A. A., Ludvig, E. A., 2013. Time representation in reinforcement learning models of the basal ganglia. Frontiers in Computational Neuroscience, 7 (194), doi: 10.3389/fncom.2013.00194.

Gigliotta, O., Nolfi, S., 2008. On the coupling between agent internal and agent/environmental dynamics: Development of spatial representations in evolving autonomous robots. Adaptive Behavior, 16 (2-3), 148–165.

Giles, C. L., Griffin, R. D., Maxwell, T., 1988. Encoding geometric invariances in higher-order neural networks. In: Anderson, D. Z. (Ed.), Neural Information Processing Systems (NIPS 1987). American Institute of Physics, New York, NY, pp. 301–309.

Giles, C. L., Maxwell, T., 1987. Learning, invariance, and generalization in high-order neural networks. Applied Optics, 26 (23), 4972–4978.

Gisiger, T., Boukadoum, M., 2011. Mechanisms gating the flow of information in the cortex: what they might look like and what their uses may be. Frontiers in Computational Neuroscience, 5 (1), doi: 10.3389/fncom.2011.00001.

Gittins, J. C., 1979. Bandit processes and dynamic allocation indices. Journal of the Royal Statistical Society. Series B (Methodological), 41 (2), 148–177.

Grace, A. A., 2000. Gating of information flow within the limbic system and the pathophysiology of schizophrenia. Brain Research Reviews, 31 (2), 330–341.

Grouchy, P., D'Eleuterio, G., 2014. Evolving autonomous agent controllers as analytical mathematical models. In: Sayama, H., Rieffel, J., Risi, S., Doursat, R., Lipson, H. (Eds.), Artificial Life 14: Proceedings of the Fourteenth International Conference on the Synthesis and Simulation of Living Systems. MIT Press, Cambridge, MA, pp. 681–688.

Gruber, A. J., Hussain, R. J., O'Donnell, P., 2009. The nucleus accumbens: A switchboard for goal-directed behaviors. PloS ONE, 4 (4), e5062, doi: 10.1371/journal.pone.0005062.

Hecht-Nielsen, R., 1995. Replicator neural networks for universal optimal source coding. Science, 269, 1860–1863.

Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural Computation, 9 (8), 1735–1780.

Howard, G., Bull, L., de Lacy Costello, B., Gale, E., Adamatzky, A., 2014. Evolving spiking networks with variable resistive memories. Evolutionary Computation, 22 (1), 79–103.

Huizinga, J., Clune, J., Mouret, J.-B., 2014. Evolving neural networks that are both modular and regular: HyperNEAT plus the connection cost technique. In: Proceedings of the 2014 Conference on Genetic and Evolutionary Computation. GECCO '14. ACM, New York, NY, pp. 697–704.

Husbands, P., 1998. Evolving robot behaviours with diffusing gas networks. In: Husbands, P., Meyer, J. (Eds.), Proceedings of the First European Workshop on Evolutionary Robotics (EvoRobot98). Vol. 1468 of Lecture Notes in Computer Science. Springer, Berlin, pp. 71–86.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., Hinton, G. E., 1991. Adaptive mixtures of local experts. Neural Computation, 3 (1), 79–87.

Jakobi, N., 1997. Evolutionary robotics and the radical envelope-of-noise hypothesis. Adaptive Behavior, 6 (2), 325–368.

Jordan, M. I., Jacobs, R. A., 1994. Hierarchical mixtures of experts and the EM algorithm. Neural Computation, 6 (2), 181–214.

Kandel, E., Tauc, L., 1965. Heterosynaptic facilitation in neurones of the abdominal ganglion of Aplysia depilans. The Journal of Physiology, 181 (1), 1–27.

Katz, P. S., 1999. Beyond neurotransmission: Neuromodulation and its importance for information processing. Oxford University Press, Oxford, UK.

Katz, P. S., 2003. Synaptic gating: the potential to open closed doors. Current Biology, 13 (14), R554–R556.

Katz, P. S., Frost, W. N., 1996. Intrinsic neuromodulation: altering neuronal circuits from within. Trends in Neurosciences, 19 (2), 54–61.

Kim, D., 2004. Evolving internal memory for T-maze tasks in noisy environments. Connection Science, 16 (3), 183–210.

Koutník, J., Greff, K., Gomez, F. J., Schmidhuber, J., 2014. A clockwork RNN. In: Xing, E. P., Jebara, T. (Eds.), Proceedings of the 31th International Conference on Machine Learning (ICML 2014). Vol. 32 of JMLR Workshop and Conference Proceedings. JMLR.org, pp. 1863–1871.

Lapicque, L., 1907. Recherches quantitatives sur l'excitation électrique des nerfs traitée comme une polarisation. Journal de Physiologie et de Pathologie Générale, 9 (1), 620–635.

Leerink, L. R., Giles, C. L., Horne, B. G., Jabri, M. A., 1995. Learning with product units. In: Tesauro, G., Touretzky, D. S., Leen, T. K. (Eds.), Advances in Neural Information Processing Systems 7 (NIPS 1994). MIT Press, Cambridge, MA, pp. 537–544.

Lehman, J., Miikkulainen, R., 2014. Overcoming deception in evolution of cognitive behaviors. In: Proceedings of the 2014 Conference on Genetic and Evolutionary Computation. GECCO '14. ACM, New York, NY, pp. 185–192.

Lehman, J., Stanley, K. O., 2008. Exploiting open-endedness to solve problems through the search for novelty. In: Bullock, S., Noble, J., Watson, R. A., Bedau, M. A. (Eds.), Artificial Life XI: Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems. MIT Press, Cambridge, MA, pp. 329–336.

Lehman, J., Stanley, K. O., 2011. Abandoning objectives: Evolution through the search for novelty alone. Evolutionary Computation, 19 (2), 189–223.

Linåker, F., Jacobsson, H., 2001a. Learning Delayed Response Tasks Through Unsupervised Event Extraction. International Journal of Computational Intelligence and Applications, 1 (4), 413–426.

Linåker, F., Jacobsson, H., 2001b. Mobile robot learning of delayed response tasks through event extraction: A solution to the road sign problem and beyond. In: Nebel, B. (Ed.), Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI 2001). Morgan Kaufmann, San Francisco, CA, pp. 777–782.

Littman, M. L., 2009. A tutorial on partially observable markov decision processes. Journal of Mathematical Psychology, 53 (3), 119–125.

Marder, E., Thirumalai, V., 2002. Cellular, synaptic and network effects of neuromodulation. Neural Networks, 15 (4), 479–493.

Markram, H., Lübke, J., Frotscher, M., Sakmann, B., 1997. Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs. Science, 275 (5297), 213–215.

Martin, S., Grimwood, P., Morris, R., 2000. Synaptic plasticity and memory: an evaluation of the hypothesis. Annual Review of Neuroscience, 23 (1), 649–711.

Miall, R. C., Wolpert, D. M., 1996. Forward models for physiological motor control. Neural Networks, 9 (8), 1265–1279.

Montague, P. R., Dayan, P., Sejnowski, T. J., 1996. A framework for mesencephalic dopamine systems based on predictive Hebbian learning. Journal of Neuroscience, 16 (5), 1936–1947.

Mouret, J.-B., Doncieux, S., 2012. Encouraging behavioral diversity in evolutionary robotics: An empirical study. Evolutionary Computation, 20 (1), 91–133.

Mouret, J.-B., Tonelli, P., 2014. Artificial evolution of plastic neural networks: a few key concepts. In: Kowaliw, T., Bredeche, N., Doursat, R. (Eds.), Growing Adaptive Machines. Vol. 557 of Studies in Computational Intelligence. Springer, Berlin, pp. 251–261.

Nogueira, B., Lenon, Y., Eduardo, C., Vidal, C. A., Cavalcante Neto, J. B., 2013. Evolving plastic neuromodulated networks for behavior emergence of autonomous virtual characters. In: Liò, P., Miglino, O., Nicosia, G., Nolfi, S., Pavone, M. (Eds.), Advances in Artificial Life, ECAL 2013: Proceedings of the Twelfth European Conference on the Synthesis and Simulation of Living Systems. MIT Press, Cambridge, MA, pp. 577–584.

O'Donnell, P., Grace, A. A., 1995. Synaptic interactions among excitatory afferents to nucleus accumbens neurons: hippocampal gating of prefrontal cortical input. Journal of Neuroscience, 15 (5), 3622–3639.

Ollion, C., Pinville, T., Doncieux, S., 2012a. Emergence of memory in neuroevolution: Impact of selection pressures. In: Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation. GECCO '12. ACM, New York, NY, pp. 369–372.

Ollion, C., Pinville, T., Stéphane, D., 2012b. With a little help from selection pressures: evolution of memory in robot controllers. In: Adami, C., Bryson, D. M., Ofria, C., Pennock, R. T. (Eds.), Artificial Life 13: Proceedings of the Thirteenth International Conference on the Simulation and Synthesis of Living Systems. MIT Press, Cambridge, MA, pp. 407–414.

Olshausen, B. A., Anderson, C. H., Van Essen, D. C., 1993. A neurobiological model of visual attention and invariant pattern recognition based on dynamic routing of information. Journal of Neuroscience, 13 (11), 4700–4719.

Poirazi, P., Brannon, T., Mel, B. W., 2003. Pyramidal neuron as two-layer neural network. Neuron, 37 (6), 989–999.

Redgrave, P., Prescott, T. J., Gurney, K., 1999. The basal ganglia: a vertebrate solution to the selection problem? Neuroscience, 89 (4), 1009–1023.

Rempis, C. W., 2007. Short-term memory structures in additive recurrent neural networks. Master's thesis, Bonn-Rhein-Sieg University of Applied Sciences, Germany.

Risi, S., Hughes, C. E., Stanley, K. O., 2010. Evolving plastic neural networks with novelty search. Adaptive Behavior, 18 (6), 470–491.

Risi, S., Stanley, K. O., 2010. Indirectly encoding neural plasticity as a pattern of local rules. In: Doncieux, S., Girard, B., Guillot, A., Hallam, J., Meyer, J.-A., Mouret, J.-B. (Eds.), From Animals to Animats 11: Proceedings of the 11th International Conference on Simulation of Adaptive Behavior. Vol. 6226 of Lecture Notes in Computer Science. Springer, Berlin, pp. 533–543.

Risi, S., Stanley, K. O., 2012. A unified approach to evolving plasticity and neural geometry. In: Proceedings of the 2012 International Joint Conference on Neural Networks (IJCNN 2012). IEEE, Piscataway, NJ, doi: 10.1109/IJCNN.2012.6252826.

Risi, S., Vanderbleek, S. D., Hughes, C. E., Stanley, K. O., 2009. How novelty search escapes the deceptive trap of learning to learn. In: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation. GECCO '09. ACM, New York, NY, pp. 153–160.

Robbins, H., 1952. Some aspects of the sequential design of experiments. Bulletin of the American Mathematics Society, 58, 527–535.

Rumelhart, D. E., Hinton, G. E., Williams, R. J., 1986a. Learning internal representations by error propagation. In: Rumelhart, D. E., McClelland, J. L., the PDP Research Group (Eds.), Parallel distributed processing: Explorations in the microstructure of cognition. Volume 1. MIT Press, Cambridge, MA, Ch. 8, pp. 318–362.

Rumelhart, D. E., Hinton, G. E., Williams, R. J., 1986b. Learning representations by back-propagating errors. Nature, 323, 533–536.

Russell, S., Norvig, P., 2003. Artificial Intelligence: A Modern Approach (Second Edition). Prentice Hall, Upper Saddle River, NJ.

Rvachev, M. M., 2013. Neuron as a reward-modulated combinatorial switch and a model of learning behavior. Neural Networks, 46, 62–74.

Rylatt, R., Czarnecki, C., 2000. Embedding connectionist autonomous agents in time: The 'road sign problem'. Neural Processing Letters, 12 (2), 145–158.

Schmidhuber, J., 1991. A possibility for implementing curiosity and boredom in model-building neural controllers. In: Meyer, J.-A., Wilson, S. W.

(Eds.), From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior. MIT Press/Bradford Books, Cambridge, MA, pp. 222–227.

Schultz, W., 1998. Predictive reward signal of dopamine neurons. Journal of Neurophysiology, 80 (1), 1–27.

Schultz, W., Dayan, P., Montague, P. R., 1997. A neural substrate of prediction and reward. Science, 275 (5306), 1593–1599.

Sher, G. I., 2012. Handbook of Neuroevolution through Erlang. Springer, New York, NY.

Shin, Y., Ghosh, J., 1991. The pi-sigma network: An efficient higher-order neural network for pattern classification and function approximation. In: Proceedings of the 1991 International Joint Conference on Neural Networks (IJCNN 1991). IEEE, Piscataway, NJ, pp. 13–18.

Silva, F., Duarte, M., Oliveira, S. M., Correia, L., Christensen, A. L., 2014. The case for engineering the evolution of robot controllers. In: Sayama, H., Rieffel, J., Risi, S., Doursat, R., Lipson, H. (Eds.), Artificial Life 14: Proceedings of the Fourteenth Conference on the Synthesis and Simulation of Living Systems. MIT Press, Cambridge, MA, pp. 703–710.

Silva, F., Urbano, P., Christensen, A., 2012. Adaptation of robot behaviour through online evolution and neuromodulated learning. In: Pavn, J., Duque-Mndez, N., Fuentes-Fernndez, R. (Eds.), Advances in Artificial Intelligence - IBERAMIA 2012. Vol. 7637 of Lecture Notes in Computer Science. Springer, Berlin, pp. 300–309.

Skinner, B. F., 1938. The behavior of organisms: An experimental analysis. Appleton-Century, New York, NY.

Soltoggio, A., 2008. Neuromodulation increases decision speed in dynamic environments. In: Schlesinger, M., Berthouze, L., Balkenius, C. (Eds.), Proceedings of the Eighth International Conference on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems. No. 139 in Lund University Cognitive Studies. LUCS, Lund, Sweden, pp. 119–126.

Soltoggio, A., Bullinaria, J. A., Mattiussi, C., Dürr, P., Floreano, D., 2008. Evolutionary advantages of neuromodulated plasticity in dynamic, reward-based scenarios. In: Bullock, S., Noble, J., Watson, R. A., Bedau, M. A. (Eds.), Artificial Life XI: Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems. MIT Press, Cambridge, MA, pp. 569–576.

Soltoggio, A., Dürr, P., Mattiussi, C., Floreano, D., 2007. Evolving neuromodulatory topologies for reinforcement learning-like problems. In: Proceedings of the 2007 IEEE Congress on Evolutionary Computation (CEC 2007). IEEE, Piscataway, NJ, pp. 2471–2478.

Soltoggio, A., Jones, B., 2009. Novelty of behaviour as a basis for the neuro-evolution of operant reward learning. In: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation. GECCO '09. ACM, New York, NY, pp. 169–176.

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

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., 2014. Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 15 (1), 1929–1958.

Srivastava, R. K., Greff, K., Schmidhuber, J., 2015. Highway networks. CoRR, abs/1505.00387. [online at http://arxiv.org/abs/1505.00387, accessed 20 August 2015]

Stanley, K. O., 2007. Compositional pattern producing networks: A novel abstraction of development. Genetic Programming and Evolvable Machines, 8 (2), 131–162.

Stanley, K. O., D'Ambrosio, D. B., Gauci, J., 2009. A hypercube-based encoding for evolving large-scale neural networks. Artificial Life, 15 (2), 185–212.

Sutton, R. S., Barto, A. G., 1990. Time-derivative models of Pavlovian reinforcement. In: Gabriel, M. R., Moore, J. W. (Eds.), Learning and Computational Neuroscience: Foundations of Adaptive Networks. MIT Press, Cambdridge, MA, pp. 497–537.

Sutton, R. S., Barto, A. G., 1998. Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA.

Sutton, R. S., Modayil, J., Delp, M., Degris, T., Pilarski, P. M., White, A., Precup, D., 2011. Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In: Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 2. AAMAS '11. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, pp. 761–768.

Thorndike, E. L., 1911. Animal Intelligence. Macmillan, New York, NY.

Tonelli, P., Mouret, J.-B., 2011a. On the relationships between synaptic plasticity and generative systems. In: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation. GECCO '11. ACM, New York, NY, pp. 1531–1538.

Tonelli, P., Mouret, J.-B., 2011b. Using a map-based encoding to evolve plastic neural networks. In: Proceedings of the 2011 IEEE Workshop on Evolving and Adaptive Intelligent Systems (EAIS 2011). IEEE, Piscataway, NJ, pp. 9–16.

Tonelli, P., 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, doi: 10.1371/journal.pone.0079138.

Triesch, J., 2007. Synergies between intrinsic and synaptic plasticity mechanisms. Neural Computation, 19 (4), 885–909.

Tuckwell, H. C., 1988. Introduction to Theoretical Neurobiology: Volume 1, Linear Cable Theory and Dendritic Structure. Cambridge University Press, Cambridge, UK.

Ulbricht, C., 1996. Handling time-warped sequences with neural networks. In: Maes, P., Mataric, M. J., Meyer, J.-A., Pollack, J., Wilson, S. W. (Eds.), From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior. MIT Press, Cambridge, MA, pp. 180–189.

Usher, M., Cohen, J. D., Servan-Schreiber, D., Rajkowski, J., Aston-Jones, G., 1999. The role of locus coeruleus in the regulation of cognitive performance. Science, 283 (5401), 549–554.

Vassiliades, V., Christodoulou, C., 2013. Toward nonlinear local reinforcement learning rules through neuroevolution. Neural Computation, 25 (11), 3020–3043.

Verbancsics, P., Stanley, K. O., 2011. Constraining connectivity to encourage modularity in HyperNEAT. In: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation. GECCO '11. ACM, New York, NY, pp. 1483–1490.

Vogels, T. P., Abbott, L. F., 2005. Signal propagation and logic gating in networks of integrate-and-fire neurons. Journal of Neuroscience, 25 (46), 10786–10795.

Williams, G. V., Goldman-Rakic, P. S., 1995. Modulation of memory fields by dopamine D1 receptors in prefrontal cortex. Nature, 375 (6541), 572–575.

Yamauchi, B. M., Beer, R. D., 1994. Sequential behavior and learning in evolved dynamical neural networks. Adaptive Behavior, 2 (3), 219–246.

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

Yoder, J., Yaeger, L., 2014. Evaluating topological models of neuromodulation in polyworld. In: Sayama, H., Rieffel, J., Risi, S., Doursat, R., Lipson, H. (Eds.), Artificial Life 14: Proceedings of the Fourteenth International Conference on the Synthesis and Simulation of Living Systems. MIT Press, Cambridge, MA, pp. 916–923.

Ziemke, T., Bergfeldt, N., Buason, G., Susi, T., Svensson, H., 2004. Evolving cognitive scaffolding and environment adaptation: a new research direction for evolutionary robotics. Connection Science, 16 (4), 339–350.

Ziemke, T., Thieme, M., 2002. Neuromodulation of reactive sensorimotor mappings as a short-term memory mechanism in delayed response tasks. Adaptive Behavior, 10 (3-4), 185–199.