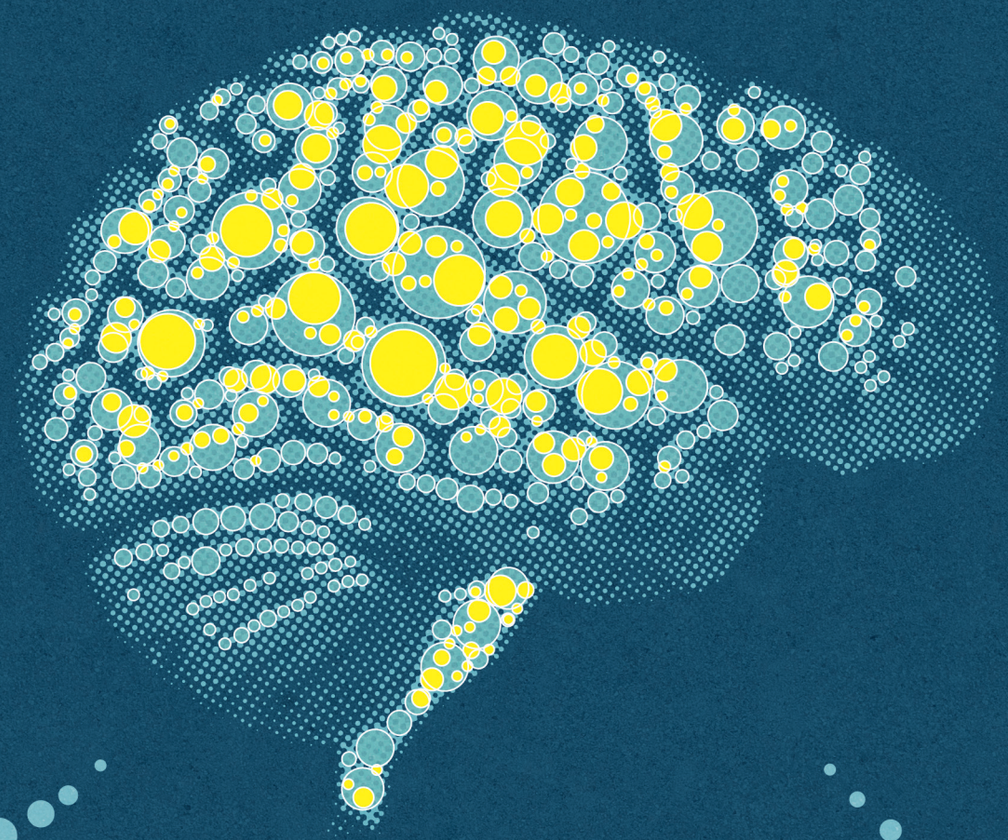
Biologically Plausible Reinforcement Learning

Jaldert Rombouts

BIOLOGICALLY PLAUSIBLE REINFORCEMENT LEARNING



Jaldert Rombouts

VRIJE UNIVERSITEIT

Biologically Plausible Reinforcement Learning

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad Doctor aan
de Vrije Universiteit Amsterdam,
op gezag van de rector magnificus
prof. dr. F.A. van der Duyn Schouten,
in het openbaar te verdedigen
ten overstaan van de promotiecommissie
van de Faculteit der Aard- en Levenswetenschappen
op vrijdag 4 september 2015 om 11.45 uur
in de aula van de universiteit,
De Boelelaan 1105

door

Jaldert Olaf Rombouts

geboren te Groningen

promotor: prof.dr. P.R. Roelfsema
copromotor: dr. S. M. Bohte

Cover design Bouwe van der Molen
Printed by Gildeprint, Enschede

Contents

Acknowledgements

"The good life is one inspired by love and guided by knowledge."
— Bertrand Russell (1872-1970)

The research presented in this thesis is the result of more than four years of work. In those years I have had the pleasure of working with smart colleagues and making friendships with many wonderful people that supported me along the way.

First and foremost I want to thank Sander for giving me the opportunity to pursue a PhD degree, and for the fruitful work (and non-work!) discussions we had over the years. In addition, I am very grateful that I could work with Pieter, who agreed to be my promotor and taught me a lot about doing and presenting science. His drive and eye for detail are an inspiration.

Centrum Wiskunde en Informatica (CWI) is a great and supportive place, and I was lucky to have such wonderful colleagues. I always enjoyed having lunch together, and of course the many nice outings that we had. In particular I enjoyed the after-lunch table soccer matches with Gunnar, Mohammed, Tobias, Davide, Joana, Raf, Sonja, Louis and Murray. Of course this list would not be complete without Margriet, whose hard work was always motivating to me. CWI is a wonderful place for science, in part because of its excellent supporting staff. I would like to thank Karin, Nada and Maarten in particular for always helping me out.

I would also like to thank the Jollemanhof 'gang'; René, Ana, Victor, Rafaela, Carolina and José. I will always remember the great time we had together, and I look forward to continue the tradition of visiting each other all over the world! Further, I thank Stephanie, Matthijs, Thomas, Bouwe (who also designed the beautiful cover of this book), Holger, Anja, Wout, Lia, Erik, Madelon, Marieke and Wolf for their friendship and support over the years.

I would like to especially thank Halldóra (for her love of life, daily conversations and her concerts) and René (for all the dinners, his thoughtfulness and his inspiring enterprises) for being my paranymphs, a true honor.

I gratefully acknowledge a grant from the Netherlands Organization for Scientific Research (grant 612.066.826), without which all of this work would not have been possible.

My sincere thanks also go out to my reading committee: Robert Babuska, Peter Dayan, Gusz Eiben, Bert Kappen, and Huibert Mansvelder. Thank you for taking the time to read my thesis, for your valuable comments, and of course for attending my defense.

I would not have been where I am today without my parents, Winand and Annemarie; thank you for your unconditional and limitless trust and support. Finally, I want to thank Tessa. Thank you for your support, for your patience while I worked on my thesis, and for being my loving companion ever since our journey started at university.

1 Introduction

"All models are wrong, but some are useful."
— George E P Box (1919-2013)

The brain is a very impressive organ. It is made up of an estimated 85 billion neurons (Azevedo et al., 2009) interconnected by 100 – 500 trillion synapses. Neurons communicate with each other by sending electrical impulses, called spikes, that influence the receiving neuron depending on the synaptic strength of the connection[1]. It is widely believed that the connection pattern between neurons and the synaptic strengths represent everything that you know about the world.

One of the most useful properties of the brain is that it can learn new ideas and skills, such as riding a bike or calculus. Learning involves changing the connection patterns and synaptic strengths in the brain. If you ever had a pet dog, you know you can train animals to do tricks, just by rewarding them for desired behavior. Somehow, by giving rewards, we can rewire the brains of animals to perform interesting behaviors (though see Breland and Breland, 1961). With this thesis I hope to contribute to the understanding this type of learning. The work in this thesis builds on ideas in animal learning, the mathematical framework of reinforcement learning, neuroscience and artificial neural networks. In the remainder of this chapter I will give a birds-eye overview of the context in which this thesis fits, and in the last section I will give an outline of the thesis.

[1] This holds for chemical synapses, which are the most common in the brain (Gerstner and Kistler, 2002).

1.1 Animal Learning

The scientific study of animal learning started at the beginning of the 20th century with the work of Ivan Pavlov and Edward Thorndike. Pavlov is most well known for his conditioning experiments with dogs (Pavlov, 1928). He trained dogs to associate a stimulus (e.g. a bell) with receiving a bite of food a short while later. Normally, dogs start to salivate when they are presented with food, but after a number of co-occurrences of the bell and the food, the dogs would start salivating at the sound of the bell. The physical process of salivation thus showed that the dog 'expected' to receive food after hearing the bell; a conditioned reflex. Learning to associate a stimulus with a reward or punishment in this way is called classical or Pavlovian conditioning.

Although Thorndike is less well known than Pavlov, his work was both earlier and perhaps more general. One limitation of the work of Pavlov is that animals did not need to *do* anything in order to receive their rewards, while in the real world animals need to perform actions to achieve their goals. Thorndike devised puzzle boxes in which he placed animals that had to find a way to escape from the box, for instance by pressing a button or pulling a lever (Thorndike, 1898). When the animal finally escaped, it was rewarded with food. By plotting the amount of time that it took animals to escape from the box over subsequent trials, Thorndike could show that learning typically followed an 'S' shape—animals were slow to escape in the beginning, then 'discovered' the trick and got faster and faster at escaping the box, and finally the escape times reached a minimum. Based on these experiments he formulated a theory of learning. The two main elements of this theory were that animals learn by trial-and-error, and that behavior follows the *law of effect*—actions that tend to be followed by a positive outcome become more likely and actions that tend to be followed by negative outcomes become less likely to be executed by the animal. Thorndike also seems to have coined the term *reinforcement*: strengthening a certain behavior by giving rewards. The type of learning that Thorndike studied is now called instrumental or operant conditioning, contrasting with the classical conditioning studied by Pavlov. In general both classical and operant conditioning are forms of *associative* learning.

Reinforcement

The experiments above make clear that animals can learn associations between stimuli and sequences of actions that lead to rewarding outcomes, but the theories that were developed were *word theories*. The problem with word theories is that they are *vague*. For instance, what exactly does the law of effect predict? How does the probability of executing an action (e.g. pressing a button) change after the animal successfully escapes from the

puzzle box after pressing this button? Word theories are fundamentally different from the mathematical theories that allow precise predictions in physics, e.g. it is possible to predict quite precisely how much time it takes for a ball to fall to the ground from any specified height quite accurately, given that we know the relevant parameters (mass, size, spin of the ball, atmospheric conditions, etc.). In the 20th century various mathematical models of learning were developed, e.g. in the branch of mathematical psychology with prominent researchers such as Hull, Bush and Mosteller (see e.g. Lovie, 2005; Niv, 2009, and references therein for a more detailed history)—the most pertinent to our discussion is the framework of Reinforcement Learning (Sutton and Barto, 1998) to which I will turn next.

1.2 Reinforcement learning

Reinforcement Learning (RL)

The aim of the Reinforcement Learning (RL, Sutton and Barto, 1998) framework is to understand how intelligent agents can learn to achieve goals while interacting with a (possibly stochastic) environment. The explicit sequential nature of the learning problem is what sets RL apart from other problems in machine learning, such as classification and supervised learning. Learning in the RL framework is very similar to animal learning; the algorithms learn by interacting with their environment in a trial-and-error fashion. Under certain conditions it can be guaranteed that the algorithms will learn the optimal sequence of actions, where optimality is defined as a function of the total possible reward that the agent manages to extract from the environment.

The mathematical framework of Reinforcement Learning is based on the experimental results of animal learning and on the framework of dynamic programming from optimal control theory (Bellman, 1957). Sutton and Barto (1998) trace the roots of computational investigations into trial-and-error learning to work by Minsky (1954) and Farley and Clark (1954) in the 1950s, through work by Klopf in the 1980's. A very readable account of the historical background of RL can be found in (Sutton and Barto, 1998). A good review that focusses on the links between neuroscience and RL is Niv (2009). In the next chapter I will formally introduce the RL framework, focussing on the concepts relevant to this thesis.

Temporal difference learning and the reward-prediction error theory of dopamine

Temporal Difference (TD) learning

One of the most powerful ideas in RL is the concept of temporal difference

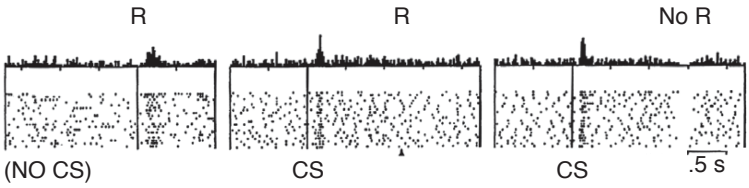


Figure 1.1: *Dopamine neurons code for reward prediction errors. Left, an unexpected reward (R) occurs. Middle, a reward occurs after a conditioned stimulus (CS). Right, no reward occurs after the conditioned stimulus. The graphs show peristimulus time histograms (top) of spike rates of a single dopamine neuron over multiple trials (bottom). The figure is adapted from Schultz, Dayan, and Montague (1997).*

(TD) learning. TD learning is powerful because it allows agents to learn rewarding behaviors purely by interacting with their environment, in contrast to for instance dynamic programming techniques that require detailed knowledge about the environment. A second important property is that TD learning methods learn by bootstrapping—they learn to improve their own estimates by experience, by building on their own earlier estimates.

The RL framework is agnostic about the learning agent—it could be a brain, but it could also be a set of tables in computer memory. It also does not specify how the learning agent actually implements the algorithms. It was therefore a groundbreaking discovery that certain neurons seem to encode exactly the temporal difference error (Montague, Dayan, and Sejnowski, 1996; Schultz, Dayan, and Montague, 1997) I discussed above.

Dopamine neuron

These midbrain (or, mesencephalic) *dopamine neurons*[2] in the substantia nigra and ventral tegmental area (VTA) project diffusely throughout the brain, and can thus inform large areas of the cortex about temporal difference errors.

Figure 1.1 illustrates the prediction error behavior of dopamine neurons under various conditions. In the left panel, an untrained monkey receives a reward (a drop of fruit juice) in the absence of any prediction, leading to a positive prediction error. The dopamine neuron shows a burst of activity on the delivery of this unexpected reward. The middle panel of Figure 1.1 shows the behavior of a dopamine neuron after the animal is

[2]Dopamine neurons are neurons that release the neurotransitter dopamine. Neurotransmitters are chemicals that can transmit information from one neuron to another. When I refer to dopamine, I always refer to brain dopamine.

trained to associate a stimulus with a reward that is delivered slightly later (i.e. a conditioned stimulus)—the stimulus thus predicts the reward. Dopamine neurons now burst in response to the onset of the stimulus, and not when the reward is actually delivered. Why? Clearly, the arrival of the conditioned stimulus predicts that a drop of fruit juice will arrive shortly, so this gives a positive prediction error[3]. The reward, which occurs exactly as predicted, does not. Finally, when the trained monkey is presented with the reward predicting stimulus but the reward is withheld, the dopamine neuron shows a dip relative to its baseline activation at the time that the reward was expected (Figure 1.1, right): there is a negative prediction error because the expected reward did not materialize. These patterns also hold for instrumental tasks, where the animals have to perform actions in order to get their reward (e.g. Schultz, Apicella, and Ljungberg, 1993).

The temporal difference learning interpretation of dopamine signaling establishes a bridge between animal learning and the theoretical framework of reinforcement learning, and it is one of the greatest successes of computational neuroscience to date.

Dopamine can influence the synaptic plasticity between neurons, and in this way the global reward prediction error can influence learning (see e.g. Wise, 2004). However, the dopamine signal in itself is not enough to explain learning: synaptic plasticity needs to be constrained to those synapses that were actually responsible for the observed reward prediction errors; the models we developed in this thesis offer one possible mechanism for how this could be implemented in the brain.

Although I emphasized the temporal difference interpretation of dopamine in this thesis, there is significant evidence that dopamine has more functions. For instance, firing rates of midbrain dopamine neurons also seem to be influenced by motivation (Dayan and Balleine, 2002; Satoh et al., 2003), reflect salient stimuli (Horvitz, 2000; Redgrave, Prescott, and Gurney, 1999), reward uncertainty (Fiorillo, 2003), and the start of new trials (Bromberg-Martin, Matsumoto, and Hikosaka, 2010a, although these results might be explained by TD given timing uncertainty). Some authors have even suggested different interpretations of the DA signal, for instance Redgrave and Gurney (2006) suggest it has an important role in the discovery of new actions. Further, working memory representations in prefrontal cortex have been shown to be sensitive to dopamine levels (Vijayraghavan et al., 2007; Williams and Goldman-Rakic, 1995), implying that dopamine plays a more direct role in working memory tasks than just

[3] Please see secion 2.1 in the next chapter for a mathematical description of TD learning.

influencing synaptic plasticity as I assumed in this thesis.

Dopamine is also not the only neurotransmitter that is associated with learning and signaling rewards. Other neurotransmitters that seem to be involved are Acetylcholine (ACh, Kilgard and Merzenich (1998) and Richardson and DeLong (1986)) and Norepinephrine (NE). One interpretation is that these neurotransmitters are important for informing the brain about 'expected' and 'unexpected' uncertainty (Yu and Dayan, 2005). These functions are related to model-based reinforcement learning (see section 2.1), which is beyond the scope of this thesis, although I think it will be extremely worthwhile to combine such methods with ours. Very recent work seems to indicate that 5-HT (Serotonin) and Glutamate might also be involved in signaling rewards (Liu et al., 2014).

The agents in this thesis consisted of simple rate-coding artificial neural networks, and the models we developed demonstrate how such networks can be trained by reinforcement learning, combining ideas from biology, artificial neural networks and RL. In the next section I briefly discuss the history of artificial neural networks and then discuss why we chose this model.

1.3 Artificial Neural Networks

Barto and Sutton (Barto, Sutton, and Brouwer, 1981) and others (Rumelhart, Hinton, and Williams, 1986) also revived the idea that networks of 'artificial' neurons might be relevant for developing artificial intelligence.

The first artificial neural networks were developed in the 1940's by McCulloch and Pitts (1943) and later by Widrow and Hoff (1960) and Rosenblatt (1962). These networks were loosely inspired by the brain, and consist of simple binary units[4] called perceptrons[5] that sum the inputs they receive, weighted by connection strengths (synapses) and 'fire' when the input exceeded a threshold. While there was initially a lot of excitement about these models, they were found to be fundamentally limited (Minsky and Papert, 1969).

The general belief is that Minsky and Papert (1969) showed that single layer neural networks[6] are unable to solve problems that require non-linear transformations, such as the famous exclusive OR problem, but this is in fact not true (see e.g. Bishop, 1995). If perceptrons receive inputs

[4] I use the term 'units' to refer to model neurons, so that 'neuron' is reserved for the biological variety.

[5] Also known as semi-linear, linear threshold, adaLines, or McCulloch-Pitts units.

[6] In this thesis I use the term layer to refer to a layer of plastic connection weights.

from an carefully selected set of fixed pre-processing units (or, basis functions) it can be shown that they can solve any classification problem[7]. What Minsky and Papert actually demonstrated is that single layer networks with pre-determined and fixed pre-processing units cannot be guaranteed to solve any problem without an extremely large set of such pre-processing units (exponential in the dimension of the problem). This severely limits the practical value of single layer perceptrons. Furthermore, although the theoretical results in the book are restricted to single layer networks, the authors made the strong conjecture that a quest for multi-layer learning rules would be sterile (Minsky and Papert, 1969, p. 231-232). After the publication of the book, many researchers thought it implied neural networks would never solve any real problems, and thus lost interest. The abandonment of connectionism, led, amongst other disappointments with Artificial Intelligence (such as the failure of expert systems and machine translation to live up to their hype) to a period called the 'AI-winter'.

A breakthrough came with the popularization of the error-back-propagation algorithm (Rumelhart, Hinton, and Williams, 1986), a powerful and efficient method for learning in essentially arbitrary network structures, and in specific in multi-layered networks with sigmoidal non-linear units as "neurons". Error-backpropagation is based on gradient-descent, and provides a very efficient and simple way to do the required computations. For a gradient-based method to work, small changes in synaptic weights need to result in small changes in the output of units, a condition that is not met by the threshold function of the perceptrons discussed above. Replacing discrete thresholds with smooth (differentiable) activation functions solves this problem. In the next chapter, I will discuss backpropagation in detail. Although the error-backpropagation algorithm provides us with a recipe for determining synaptic updates in networks of neuron-like elements, it has various problems as a model for learning in the brain, as I will also discuss in the next chapter.

Like the typical artificial neural networks used in machine learning, the models we developed in this thesis had two layers of plastic weights, and the units have smooth, differentiable, activation functions. This is quite

[7] Such ideas are still widely used, for instance in Support Vector Machines (Boser, Guyon, and Vapnik, 1992) and Echo/Liquid state machines (Jaeger, 2001; Maass, Natschläger, and Markram, 2002). There is even neurophysiological evidence that the brain might use random projections (Luo, Axel, and Abbott, 2010).

an abstraction from real brains—in the next section I will discuss why we chose this level of modeling.

1.4 The Level of Modeling

Biologically Plausible

A term that is used often in this thesis is *biologically plausible*, and it is fruitful to elaborate on exactly what I mean with this term. The models that are described in this thesis are all abstract rate coding networks[8], in the tradition of the neural networks that were popularized in the 1980's. These models contain many elements which are known to conflict with biology, a simple example being that synaptic weights in biological neuronal networks are either excitatory *or* inhibitory[9]—synapses have never been observed to change their 'sign'. In our models, we do however assume that connection weights can change 'sign'[10]. Another assumption is that units in our models communicate by a firing rate (the number of spikes per second), while there is ample experimental evidence that in many settings the temporal details of the spike train matter (see e.g. Gerstner and Kistler, 2002; Rieke, 1999, for an in-depth discussion).

How can we call models that go against basic neuroscientific facts biologically plausible? This depends on the level of modeling that one chooses. There is a whole continuum of different levels of modeling, from detailed molecular level models of neurons to very abstract mean field models that describe the dynamics of whole populations of neurons. One important observation was made by the statistician George E P Box (1919-2013): "*essentially, all models are wrong, but some are useful*". One can take the most detailed biophysical model that has been made to date and point out where the model does not match biology. Further, there are many details about biology which are completely unknown to us. For instance, the biological machinery of synapses is nothing short of astounding, even at the level of understanding we have now (e.g. Kandel, Schwartz, and Jessell, 2013). In short, any model that will ever be made of the brain will be in some sense biologically implausible. The key is in the 'usefulness' of models.

I believe that the chosen level of abstraction can inform us about the brain, given that some assumptions hold. The first one is that the detailed spike-by-spike level is not relevant for the tasks that we want to model and the second one is that the behavior of excitatory and inhibitory neurons can

[8] This type of model can be derived from the Hodgkin-Huxley equations (Hodgkin and Huxley, 1952) however.

[9] This is called Dale's principle, or Dale's Law.

[10] Though see section 2.2 for ideas on how this assumption can be relaxed.

be described by a unit type that allows for positive and negative connection weights and switches in sign. What I then mean by biologically plausible is that all information that is required to determine connection weight changes is available locally, at the synapse.

We are not the first who have tried to build biologically plausible neural networks that can be trained by reinforcement learning, and our approach is certainly not the most biologically detailed. What I believe sets our methods apart from other work is that they show how to train multi-layer networks by RL, while others have mostly worked on training essentially single layer networks[11] (see e.g. Potjans, Morrison, and Diesmann, 2009; Soltani and Wang, 2009; Urbanczik and Senn, 2009). While single layer networks can be useful, they are limited in the types of problems that they can solve (as was famously discovered in the context of perceptrons, see section 1.3). Multi-layered networks have the ability to learn representations that are not explicitly given to them (e.g. the concept 'chair' from the widely different examples of chairs in the world). It is a widely held belief that the brain works by building on layers of such representations to deal with the complexity of the world (see e.g. Fukushima, 1980; Serre et al., 2007). *Deep learning* machine learning approaches that use this idea are currently the state of the art (Ciresan et al., 2010; Hinton and Salakhutdinov, 2006; Le et al., 2012). Very recently, similar ideas have also been applied in the context of reinforcement learning, yielding state-of-the-art performance (and even outperforming human experts) on various ATARI games (Mnih et al., 2013). While such pure machine learning approaches are more powerful than the models we introduce here, these methods also clearly show the potential of moving beyond single layer models.

1.5 Outline of the thesis

In the next chapter I will introduce the framework of reinforcement learning, artificial neural networks and the ideas of the Attention-Gated Learning (AGREL, Roelfsema and van Ooyen, 2005) framework on which the rest of the thesis builds.

The work in this thesis was aimed at investigating whether the key ideas of AGREL could be extended to work for more general problems than the tasks described in Roelfsema and van Ooyen (2005). In chapter 3, I introduce the first generalization of the AGREL framework, MQ-AGREL.

[11] More specifically, only a single layer of plastic weights; the networks may have arbitrary hardwired (non-learned) components.

This chapter contains two new ideas; first, it replaces the action-probability learning mechanism of AGREL with an action-value learning mechanism, giving a clear interpretation of what the AGREL network tries to learn. Second, I show that attention-gated learning can be generalized to learning multiple simultaneous actions, instead of single binary actions, while learning is still driven by a scalar prediction error signal.

The AGREL and MQ-AGREL models are both limited to direct-reward reinforcement learning, whereas in most real world tasks animals need to learn sequences of actions in order to gain food or other rewards. In many real world tasks, one also needs some form of working memory, for example for choosing the right direction at an intersection based on a road-sign some hundreds of meters before. In chapter 4, I introduce the Attention Gated MEmory Tagging (AuGMEnT) model which generalizes the ideas of MQ-AGREL from direct-reward reinforcement learning to sequential reinforcement learning. The model extends beyond standard reinforcement learning approaches because it contains a set of memory cells that allow it to learn tasks that require working memory. I use the AuGMEnT model introduced to model four different working memory tasks from the neuroscience literature, and we show that the model is able to 1) learn the same tasks as monkeys and 2) that the representations that are formed in trained networks are similar to those found in monkeys when they are trained on the same tasks. In chapter 5, I investigate how AuGMEnT learns an attentional filtering task by trial-and-error learning.

An assumption we make in AuGMEnT is that there is a mechanism that informs the network that it has reached the end of a trial and that it should reset its working memory before a new trial starts. Animals learning similar tasks however need to *learn* when trials begin and end, which may not be a trivial feat. In chapter 6, I extend the AuGMEnT framework to learn to recognize trial boundaries, enabling the network to learn complete tasks purely by reinforcement learning.

2 Background

This chapter introduces the mathematical framework and notation that we use throughout the thesis. We start with the description of the Reinforcement Learning framework, then discuss Artificial Neural Networks, and finally we discuss the main ideas of the AGREL model, which forms the basis for this thesis. Those familiar with reinforcement learning (RL) and artificial neural networks can skip directly to the last paragraph of section 2.2.

2.1 Reinforcement Learning

As discussed in chapter 1, Reinforcement Learning is aimed at understanding how intelligent agents can learn to achieve goals while interacting with a (possibly stochastic) environment (Sutton and Barto, 1998). One of the basic assumptions of RL is that the goal of any animal is to maximize the total reward it can extract from the environment. It is also an assumption that all rewards (e.g. food or drinks) can be expressed as scalar values r. We give a short overview of the concepts from RL that are especially relevant in the context of this thesis, where we mostly follow Sutton and Barto (1998).

Markov Decision Process

Markov Decision Process (MDP)

The standard framework in Reinforcement Learning is called a Markov Decision Process (MDP). The interaction between an agent and its environment in an MDP is visualized in Figure 2.1. The standard RL framework assumes that time t is discrete, and that there is a finite and discrete set of world states $\mathcal{S}$ and actions $\mathcal{A}$. At every time t, the agent receives a scalar reward $r(t)$ and a representation of the world state $s(t) \in \mathcal{S}$. It responds by selecting an action $a(t) \in \mathcal{A}(s)$[1], and the environment responds to the action by emitting a new state $s(t+1)$ and a reward $r(t+1)$.

[1] This notation allows for the possibility that not all actions might be possible in state s. We will write $\mathcal{A}$ as a shorthand, with the understanding that the set is possibly limited given the state s.

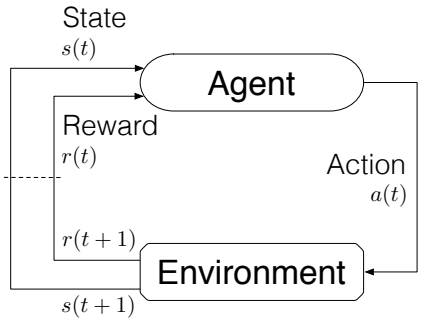


Figure 2.1: *Diagram of the interaction between an agent and its environment (task). Based on figure 3.1 in Sutton and Barto (1998)*

The transitions between states given actions are represented by the transition matrix $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0,1]$ mapping from (state,action) pairs to a probability distribution over possible successor states. The shorthand $\mathcal{P}_{sa}^{s'}$ is used to refer to the probability of transitioning from state s to s' given that action a is selected in s.

A reward matrix $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ maps any possible transition to a scalar reward r, and the shorthand $\mathcal{R}_{sa}^{s'}$ is used to refer to the expected reward (since rewards may also be stochastic) for choosing action a in state s and transitioning to state s'.

The tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$ completely defines the MDP. The word 'Markov' refers to the assumption that all the information for taking the optimal decision at any time t is contained in the state[2] $s(t)$ (to be more precise, this is first order Markov, which is default in most work). Technically, this is expressed as equality between the following conditional probability distributions:

$$\begin{aligned} &P\left[s(t+1) = s', r(t+1) = r \mid s(0), a(0), r(0), \ldots, s(t), a(t), r(t)\right] = \\ &P\left[s(t+1) = s', r(t+1) = r \mid s(t), a(t)\right] , \end{aligned} \tag{2.1}$$

for all s', r and all histories $s(0), a(0), r(0) \ldots s(t), a(t), r(t)$. This means that the state s' and reward r' at time $t+1$ only depend on the state and action at time t—the agent could not do better by having more information. In many real world problems the Markov assumption is clearly false, and in specific

[2] To avoid confusion, 'state' in standard RL refers to a signal given to the agent (Sutton and Barto, 1998), not the true hidden state of the world as in generative models or Partially Observable MDPs (see below).

it is false in the domain of working memory tasks that we will discuss from chapter 4 onwards; such tasks fall in the domain of Partially Observable MDPs.

Partially Observable MDPs In a partially observable MDP it is assumed that (some of) the state variables of the world are *hidden* or *latent*, i.e. that they are never directly observed, but have to be inferred from observations[3]. Following Kaelbling, Littman, and Cassandra (1996), a POMDP can be described as a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \Omega, \mathcal{O} \rangle$, where $\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}$ would define a standard MDP as defined in the previous section, Ω the (finite) set of observations that the agent can experience and $\mathcal{O}$ an $\mathcal{S} \times \mathcal{A} \rightarrow \Pi(\Omega)$ *observation function*, mapping from each action and *resulting* state to a probability distribution over observations.

An important construct in the POMDP framework is the *belief state*. Since the agent does not observe in which *true* world state it is, it estimates the probability that it is in any of the possible world states. This belief state should capture the sufficient statistics about the history of the agent, i.e. it should contain all the relevant information that can be extracted from the past to make optimal decisions at the current time. A component called the *state estimator* is responsible for updating the belief state given the last action, the current observation and the last belief state. This is done using Bayes' rule, which is relatively straightforward if one has access to the true underlying probabilistic model, as is often assumed[4]. Finally, a controller (policy) maps the belief state to the optimal action.

Traditionally, POMDP problems are more seen as planning problems (model-based) than as learning problems, in that it is often assumed that the agents have access to the structure of the POMDP model, and that they only need to find the optimal (action) value function (Kaelbling, Littman, and Cassandra, 1996). Bayes Adaptive (PO)MDPs (Duff, 2002; Ross, Chaib-draa, and Pineau, 2007) can additionally learn the model parameters (e.g. the transition function). More recently there has been work on learning POMDP model parameters and structure from scratch, see for instance Doshi-Velez (2009), which we believe to be an important direction in RL research.

The models for learning working memory tasks we have developed (chapter 4 and onwards) do not make use of an explicit belief state, rather,

[3] But note that MDPs are a subset of POMDPs.

[4] If you do not, there are full Bayesian formulations for both MDPs and POMDPs, called Bayes-Adaptive MDPs (BAMDPs, see Duff, 2002, and references therein) and Bayes-Adaptive POMDPs (Ross, Chaib-draa, and Pineau, 2007), respectively.

the models' task can be seen as constructing useful internal states (in the first order Markov sense) based on incoming sensations. They are aimed at a subset of the general POMDP problem, namely problems that can be solved by remembering (some) information about past observations (although a *full* history model would be equivalent to belief states). For now, we shall return to the discussion of the standard MDP framework of RL.

Episodic and infinite horizon tasks

Terminal state

It is convenient to expand the set of states $\mathcal{S}$ with a special terminal or absorbing state, which can only transition to itself with a reward of 0. The extended set of states is written $\mathcal{S}^+$. This addition is useful because it allows for the treatment of terminating, episodic (achievement/finite horizon) tasks and for non-terminating (maintenance/infinite horizon) tasks in the same framework. An example of the former is navigating though a maze, from start to end, and an example of the latter is maintaining a certain temperature in a room by controlling a heating element.

Expected future discounted reward

We can now formally state the goal of an agent as maximizing, at each time-step, the expected future discounted reward[5] (also termed *discounted return*) $\mathrm{E}[R_t]$, with R_t:

$$R_t = \sum_{k=0}^{\infty} \gamma^k r(t+k+1)\,, \tag{2.2}$$

Discount rate, γ.

where $\gamma \in [0,1]$ is called the *discount rate*. This discount rate expresses mathematically that a reward now is more valuable than the *same* reward at a later point in time. If $\gamma < 1$ and all $|r| < \infty$, the expected future discounted reward is finite. This is essential, because an agent performing an infinite horizon task with $\gamma = 1$ cannot distinguish between different actions since all yield an infinite total reward[6].

Value functions

The behavior of an agent can be summarized by its policy $\pi(a \mid s)$, which gives the probability that the agent selects action a in state s. The policy

state value

allows us to specify an important quantity, the state value $V_\pi(s)$:

[5] Note that there are alternative objective functions, such as for instance the total (i.e. non-discounted) return, or the expected reward per unit time.

[6] Note however that one could optimize the *rate* of rewards instead of the expected future discounted reward.

$$
\begin{aligned}
V_\pi(s) &= \mathrm{E}_\pi\{R_t \mid s(t) = s\}\,, \\
&= \mathrm{E}_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r(t+k+1) \mid S(t) = s\right\} \quad \text{by (2.2)}\,, \\
&= \mathrm{E}_\pi\left\{r(t+1) + \sum_{k=1}^{\infty} \gamma^k r(t+k+1) \mid S(t) = s\right\}\,, \\
&= \sum_{a \in \mathcal{A}(s)} \pi(a \mid s) \sum_{s' \in \mathcal{S}} \mathcal{P}_{sa}^{s'}\left[\mathcal{R}_{sa}^{s'} + \gamma \mathrm{E}_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r(t+k+2) \mid S(t+1) = s'\right\}\right]\,, \\
&= \sum_{a \in \mathcal{A}(s)} \pi(a \mid s) \sum_{s' \in \mathcal{S}} \mathcal{P}_{sa}^{s'}\left[\mathcal{R}_{sa}^{s'} + \gamma V_\pi(s')\right]\,, \qquad (2.3)
\end{aligned}
$$

the recursive relationship between $V_\pi(s)$ and $V_\pi(s')$ is called the Bellmann equation for $V_\pi(s)$. The state value $V_\pi(s)$ expresses how good it is for an agent to be in state s, in terms of the expected discounted return. *Bellmann equation*

In this thesis we are mostly concerned with *action* values, or Q-values, which express a similar concept but for pairs of state and actions. Formally: *Action/Q-value*

$$
\begin{aligned}
Q_\pi(s,a) &= E_\pi\{R_t \mid s(t) = s, a(t) = a\}\,, \\
&= \sum_{s' \in \mathcal{S}} \mathcal{P}_{sa}^{s'}\left[\mathcal{R}_{sa}^{s'} + \gamma \sum_{a' \in \mathcal{A}} \pi(a' \mid s') Q_\pi(s', a')\right]\,, \qquad (2.4)
\end{aligned}
$$

which can be read as the expected discounted return for selecting action a in state s, given policy π.

So far we have described value functions for a policy π, but what we are actually looking for is the optimal policy π^*. Following the optimal policy yields the highest return, which is what we assumed to be the goal of the agent. The optimal policy gives rise to optimal value functions V^* and Q^*, by replacing π by π^* in equations (2.3) and (2.4), and replacing the sums by *max* operators. It is always guaranteed that there is at least one optimal policy, although finding it by explicitly solving the Bellman optimality equation is only possible in some limited cases (Sutton and Barto, 1998)—the reason for this is the *curse of dimensionality* (Bellman, 1957): in many problems the size of the problem scales exponentially with the number of state variables. For instance, in a two dimensional maze, where each dimension is discretized into 10 parts, there are $10 \times 10 = 100$ states, and in a three dimensional maze there are already $1,000$ states. For realistic problems with many state variables the number of states is so vast that exact solutions become infeasible, even for our fastest computers. In practice, one thus tries to approximate the optimal value functions. *optimal policy*

An important observation is that *if* the agent would somehow know the Q^*-values, *then* an agent can select the optimal action sequence by just selecting the action with the highest value in each state, and it does not need to know anything else about the environment[7]. The Q-value function essentially *caches* the results of all one-step look aheads, and it is trivial to select the best action (Daw, Niv, and Dayan, 2005; Sutton and Barto, 1998).

Temporal Difference Learning

In the previous chapter, we said that temporal difference learning is one of the most central ideas in RL. Temporal difference learning is a class of methods for learning good approximations (in some cases provably optimal; Singh et al., 2000; Watkins and Dayan, 1992), to (action) value functions. The beauty is that TD methods can learn purely by interacting with the environment and observing the results; they do not need to know about the structure of the environment or rewards ($\mathcal{P}$ and $\mathcal{R}$). Such methods are thus also called *model-free* methods[8].

Model-free RL

TD methods work by *bootstrapping*; initially, the values of all states $V(s)$ are set to some uniform or random value. Then, each observed transition is used to improve the estimates of the state values, as:

$$V(s) \leftarrow V(s) + \alpha \left[\underbrace{\text{Target} - V(s)}_{\text{Prediction Error}} \right], \tag{2.5}$$

where $\leftarrow$ denotes assignment and where α is a (typically small) learning rate that determines the influence that the observed prediction error[9] has on the new estimated value of the state. Observe that if the prediction error is zero, the estimate remains unchanged.

What value should Target take? If we look at the Bellman equation for the value function (2.3) and interpret it like an update equation, we see that

[7] Except the set of actions to optimize over—automatic discovery of actions is a fascinating problem. Hierarchical RL methods can learn to compose primitive actions into coherent sequences of actions that can be selected as a unit after learning, such as in the options framework (Sutton, Precup, and Singh, 1999). There are also clear links to neuroscience, see e.g. Botvinick, Niv, and Barto (2009). One can also approach this problem from another direction, learning primitive actions from demonstration, see e.g. Niekum et al. (2013)

[8] There is another class of methods that are *model-based*. Although these models are very relevant (e.g. Daw, 2012), they are much less developed in the context of biologically plausible models and beyond the scope of this thesis.

[9] Throughout this thesis we use prediction error, temporal difference (TD) error or reward prediction error (RPE) to refer to such differences.

$V(s)$ depends on an expectation involving $\mathcal{R}_{sa}^{s'} + \gamma V_\pi(s')$. Unfortunately, we only observe a single transition to s' and a single reward r. Note however that this transition *does* in fact follow the policy π and the dynamics of the environment. If we would make the transition infinitely often and take the average of the value $r + \gamma V(s')$, it would converge to the same value. In the extreme case, we can take Target $= r + \gamma V(s')$, and let the averaging be performed by the agent on each occasion that it transitions from state s. This yields:

$$V(s) \leftarrow V(s) + \alpha \left[\underbrace{r + \gamma V(s') - V(s)}_{\text{Prediction Error}} \right]. \tag{2.6}$$

This simplification turns out to work very well in practice. Furthermore, as remarked earlier, this prediction error has been related to the behavior of dopamine neurons (Montague, Dayan, and Sejnowski, 1996; Schultz, Dayan, and Montague, 1997).

It is also possible to formulate TD learning rules for action values; the most well-known of these being the Q-learning algorithm (Watkins and Dayan, 1992): *Q-learning*

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right]. \tag{2.7}$$

Q-learning is what is known as an *off-policy* algorithm—this is because the error signal it uses for learning depends on the best available $Q(s',a')$, whereas the model's policy might select a different action in s'. One can change Q-learning into an *on-policy* method by dropping the max operator, and instead using the Q-value prediction in s' for the action a' that the model actually selects: *off-policy* *on-policy*

$$\begin{aligned} Q(s,a) &\leftarrow Q(s,a) + \alpha \left[r + \gamma Q(s',a') - Q(s,a) \right], \\ Q(s,a) &\leftarrow Q(s,a) + \alpha \delta(t), \\ \delta(t) &= r + \gamma Q(s',a') - Q(s,a), \end{aligned} \tag{2.8}$$

where we introduce $\delta(t)$ as a convenient shorthand for the prediction error in the square brackets. This is called the SARSA learning rule, due to the fact that it uses two sequential state-action pairs plus a reward (Rummery and Niranjan, 1994). Note that without careful control of exploration, SARSA and Q-learning will typically learn different solutions (see e.g. the cliff-walking example in Sutton and Barto (1998)). Intuitively, SARSA tends to *SARSA*

avoid moving to states that have low values, due to the fact that it incurs an error for the actually selected action, in contrast to Q-learning. In this thesis we use a learning rule based on SARSA from chapter 4 onwards.

Action selection

So far we have not discussed how actions should be selected by an agent. When the agent knows the optimal action-value function, the answer is simple: select the action with the highest value in each state. The problem is how to select actions when we do not know this value function, for instance at the beginning of learning a new task, or when the environment is non-stationary. The agent needs to find a good balance between exploiting the information it already has about the environment and exploring the environment for potentially even better rewards.

Two very common strategies are ϵ-greedy and softmax (or Boltzmann) exploration. In ϵ-greedy, the agent will select the highest valued action with probability $1 - \epsilon$ and a uniformly random action otherwise (where ϵ is typically small). Note that this uniform exploration does not take into account the Q-values the agent already knows, which can be disastrous if there are some actions that lead to very low rewards in comparison to other actions. The softmax rule is slightly smarter; the probability that the agent selects action a, $P(s, a)$ depends on the values for the different actions:

$$P(s,a) = \frac{\exp(Q(s,a)/\tau)}{\sum_{s,a'} \exp(Q(s,a')/\tau)}\,, \tag{2.9}$$

where τ is a *temperature* parameter that determines the influence of the values on the action probability (as $\lim_{\tau\to\infty} \exp(Q(s,a)/\tau) = 1$); typically this temperature is set high at the beginning of learning and is gradually lowered as learning progresses[10].

Both discussed action selection mechanisms are ad-hoc, since they do not exploit all the information that an agent has, for instance focussing exploration in regions that are unknown to it, and exploiting in regions that are well known. Although we believe that the development of models that try to address such issues (Bayesian RL, see e.g. Duff, 2002; Ross, Chaib-draa, and Pineau, 2007) is important, it is beyond the scope of this thesis.

[10] There is potential for numerical instability in the naive implementation of the softmax controller. This is avoided by subtracting $\max_a Q(s,a)$ from each Q-term, (see e.g. van Hasselt, 2011).

Eligibility Traces

In a standard TD learning method the value functions are only updated for each transition. Imagine that the agent is in a maze where all transitions except for the transition to the goal-state are unrewarded. Now, if a TD method starts with an action value-function that assigns 0 to all (state, action) pairs, the agent will only update the Q-value for the transition to the goal on the first trial. This information then slowly flows back through all paths that lead to the goal, one trial at a time. Eligibility traces speed up this process by putting a 'trace' $e(s,a)$ on each (state, action) pair that is visited. This trace typically has an exponential decay, e.g.:

$$e(s,a,t) \propto \sum_i \mathcal{H}(t-T_i)\exp(-(t-T_i))\,, \qquad 0 \le T_i \le t\,, \tag{2.10}$$

where t is the current time-step, T_i the time index of the ith visit to the pair (s,a) and $\mathcal{H}(x)$ the Heaviside function which equals 0 for $x<0$ and 1 for $x \ge 0$. The form shown here is called 'accumulating traces'—it is also common to use only the time T of the last visit, resulting in a method called 'replacing traces'. One way to interpret these traces is as a forward credit assignment signal: it states that a specific choice of state, action is partially responsible for going down the path that the agent is on. When a TD error occurs, the blame or credit is assigned to all (state, action) pairs in proportion to their eligibility. This can greatly speed up learning, for instance in the maze scenario described above.

It is possible to incorporate eligibility traces in the TD learning algorithms we discussed above; we show how to do this for the SARSA algorithm as this is the most relevant in the context of this thesis. After each transition, the following updates are computed:

$$Q(s,a) \leftarrow Q(s,a) + \beta e(s,a)\delta(t) \qquad \forall s,a\,, \tag{2.11}$$

$$e(s,a) \leftarrow \lambda\gamma e(s,a) + \begin{cases} 1 & \text{if} \quad s=s(t), a=a(t) \\ 0 & \text{otherwise} \end{cases} \qquad \forall s,a\,, \tag{2.12}$$

where $\lambda \in (0,1)$ is a *decay* parameter which determines how fast the eligibility traces decay. This method is called (linear) SARSA(λ). *SARSA(λ)*

Function approximation

So far we have implicitly assumed that all information, such as the Q-value function, is stored in tables with an entry for each state, action pair. This approach quickly becomes unfeasible for large problems, due to the

exponential growth of the table as a function of the number of states and actions. A table-based representation also fails to exploit any structure that might be present in the problem. In general, when states are very similar, one also expects the action values to be similar. Function approximation techniques try to represent the (relevant aspects of) the state in a smart way, reducing the number of parameters that need to be stored and learned for representing the value function. In the light of this thesis, the most relevant way to represent value functions is by (artificial) neural networks, which we will turn to next.

2.2 Artificial Neural Networks

As discussed in section 1.4, we chose to model the brains of agents at the level of abstract rate coding networks which were popularized in the 1980's by Rumelhart and coworkers (Rumelhart, Hinton, and Williams, 1986). Here we briefly introduce the problems neural networks can solve, and then discuss a simple example network architecture and the recipe to derive its learning rules, the error backpropagation algorithm.

The goal of a neural network is to map input vectors x to output vectors y. The goal can be classification or regression. In classification, the problem consists of determining the class membership of (novel) input vectors, for instance, determining whether fruit is ripe based on a vector of pixel colors from a photograph of the fruit, or, in the context of RL, determining the optimal action given a new observation. In regression, the problem is mapping input vectors to output vectors in $\mathbb{R}^n$, for instance, mapping $x_1, x_2 \rightarrow x_1 x_2$, or in the context of RL, mapping a new observation to a set of action values. Of course, the classification problem can be seen as a special case of the regression problem, i.e. it is straightforward to turn the output of a regressor into a classification, but knowing the goal of the network and adapting the architecture to the problem often simplifies the learning problem (Bishop, 1995).

Architecture

A typical neural network that can be used for regression is shown in Figure 2.2; it has a set of input units (x) on the top, a set of 'hidden' units y in the middle and a set of output units o on the bottom. Input units x_i are connected to hidden units y_j with connection weights v_{ij}, and hidden units are connected to output units o_k by weights w_{jk}. Inputs are presented to the network by clamping the activations of the input units x. The activations

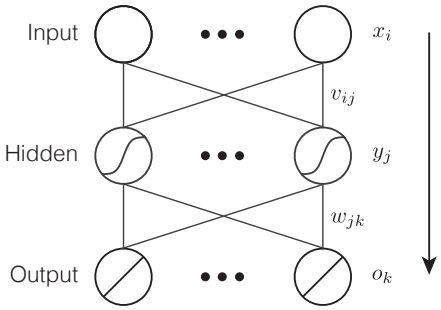


Figure 2.2: *An example neural network.*

for the hidden units y_j are then computed as:

$$a_j = \sum_i x_i v_{ij} \,, \tag{2.13}$$

$$y_j = 1/(1 + \exp(-a_j)) = \sigma(a_j) \,, \tag{2.14}$$

where a_j represents the synaptic input to unit y_j, and σ is a shorthand for the sigmoidal transformation from the input to the output of the unit. The output unit activations are computed as[11]:

$$o_k = \sum_j y_j w_{jk} \,. \tag{2.15}$$

The mapping from input vectors to output vectors is parameterized by the matrices of connection weights V and W. The next section discusses how these parameters can be learned from a set of example pairs (x, o), where the goal of the network is assumed to be regression.

Learning rules - Error Backpropagation

Here we derive the *online*[12] learning rules for the example network above, although it is straightforward to extend this to the general case. As was

[11] Note that we use just a linear activation function for the output units, as this is the form we use throughout this thesis.

[12] Traditionally, the updates were computed in batch (i.e. after presenting all training examples), but in practice it is both easier and faster to use the online form, see e.g. Wilson and Martinez (2003). In modern work people typically compromise by using mini-batches, which allow for fast parallel implementations. Recent work shows that mini-batches might even outperform the online form in some cases, see Li et al. (2014).

discussed in section 1.3, the error backpropagation algorithm is a fast[13] way to compute how connection weights in a network should change to minimize an error function, based on gradient descent.

A typical error function (or objective function) for regression tasks is the squared error:

$$E^p(t,z) = \frac{1}{2}\sum_k (t_k^p - o_k)^2 \,, \tag{2.16}$$

where t_k^p is the *target* value output unit k should take for input pattern p and o_k is the output of the network. We write E^p to emphasize that this is the error the network makes for input pattern p.

The influence that weight w_{jk} has on the observed error can be written:

$$\begin{aligned}\frac{\partial E^p}{\partial w_{jk}} &= \frac{\partial E^p}{\partial o_k}\frac{\partial o_k}{\partial w_{jk}} \,, \\ &= -(t_k^p - o_k) y_j \,,\end{aligned} \tag{2.17}$$

where the first step follows by the chain rule for derivatives and the second step follows because w_{jk} only influences the error made by unit o_k.

To derive the influence that hidden weights have on the error, we follow the same recipe:

$$\begin{aligned}\frac{\partial E^p}{\partial v_{ij}} &= \frac{\partial y_j}{\partial v_{ij}}\sum_k \frac{\partial o_k}{\partial y_j}\frac{\partial E^p}{\partial o_k} \,, \\ &= \frac{\partial a_j}{\partial v_{ij}}\frac{\partial y_j}{\partial a_j}\sum_k \frac{\partial o_k}{\partial y_j}\frac{\partial E^p}{\partial o_k} \,, \\ &= -x_i y_j (1-y_j)\sum_k w_{jk}(t_k^p - o_k) \,,\end{aligned} \tag{2.18}$$

where the sum over the k output units follows from the fact that changing the output of unit y_j influences all output units o_k, and $\frac{d\sigma(x)}{dx} = \sigma' = \sigma(x)(1-\sigma(x))$.

Now that we have computed the gradients of the error function with respect to all weights in the network we can determine the updates of the

[13] The computational complexity of the algorithm scales linearly with the number of weights, instead of quadratically, as a naive method of computing the gradients would require (Pineda, 1989).

network parameters by moving in the direction of decreasing error:

$$\begin{aligned}\Delta w_{jk} &= -\frac{\partial E^p}{\partial w_{jk}} , \\ &= (t_k^p - o_k) y_j , \\ &= \eta_k y_j , \end{aligned} \tag{2.19}$$

where η_k is a shorthand for the error $(t_k - o_k)$ made by unit k. This allows us to write the update for the hidden weights as:

$$\Delta v_{ij} = -\frac{\partial E^p}{\partial v_{ij}} = x_i y_j (1 - y_j) \sum_k w_{jk} \eta_k . \tag{2.20}$$

The η_k's flowing up through the network is why the algorithm is called error backpropagation.

Biological Plausibility

Error backpropagation provides a recipe for deriving update rules for essentially arbitrary networks, as long as the required gradients can be computed (see Bishop, 1995). Although the underlying ideas are old, going back to work on optimal control in the 1960's (see e.g. Le Cun, 1988, and references therein), it is still a main component in state-of-the-art machine learning approaches (see e.g. Ciresan et al., 2010; Hinton, 2007; Hinton and Salakhutdinov, 2006).

As a model for learning in the brain however, there are a number of problems with error backpropagation (Crick, 1989; O'Reilly, 1996; Roelfsema and van Ooyen, 2005), which we will discuss in the next paragraphs.

Teaching signal The foremost problem is that in the standard setting, it is assumed that there is a teaching signal (t_k^p in the previous section) that informs the network what the required output was. In some cases it can be argued that such a teaching signal is indeed present, for instance in motor babbling (Wolpert, Ghahramani, and Jordan, 1995) or auto-association (mapping an input to itself, potentially via a 'bottleneck' hidden layer of lower dimension (Bishop, 1995)), but in most cases this assumption does not make sense as it would imply that the agent already knows the correct mappings.

Backpropagation of errors There is neurophysiological evidence that when certain types of neurons spike, backpropagating action potentials retrogradely propagate into the dendritic tree (Stuart et al., 1997). This signal could be used as a measure of the output of a neuron, and a such could be used to influence presynaptic plasticity. The problem is that this is *not* the error signal that is required in error-backpropagation; in terms of the example network in section 2.2 what propagates back is o_k, not η_k. Another problem is that there is no evidence that this signal can propagate through multiple layers. There is also some evidence that synaptic depression induced at glutamatergic synapses can back-propagate to pre-synaptic neurons (Fitzsimonds, Song, and Poo, 1997), but this depression also does not seem to spread through multiple layers, and what spreads back is depression instead of unit-specific error signals, so it is not equivalent to error-backpropagation either. So, although there is some evidence that signals can back-propagate in biological neural networks, there is no known mechanism that could implement the backpropagation of unit-specific error signals.

Synaptic weights The BP algorithm assumes that the connection weights can change from excitatory to inhibitory, while this has never been observed experimentally. We judge this to be a less severe issue than the two other issues mentioned above, since we believe that it could be possible to change our implementations to respect Dale's law; for instance, one could potentially constrain the weights to be positive, as in O'Reilly and Frank (2006). Alternatively, it is possible to adapt networks with mixed weight units into ones with excitatory and inhibitory units that have only positive weights (Parisien, Anderson, and Eliasmith, 2008), and it might be possible to generalize such techniques. This argument cannot be made for the other issues.

In the final section of this chapter we will discuss the Attention Gated Reinforcement Learning (AGREL) model by Roelfsema and van Ooyen (2005), which is a learning scheme that has the attractive properties of error-backpropagation, but that is more biologically plausible; the ideas of AGREL formed the basis for the work in this thesis.

2.3 Attention Gated Reinforcement Learning

Attention Gated Reinforcement Learning is a biologically plausible reinforcement learning scheme for training neural networks in a direct

reward setting (Roelfsema and van Ooyen, 2005). In the direct reward setting[14], rewards are delivered immediately when the agent makes its choice, so that there is *no* dependence on time. The main result in Roelfsema and van Ooyen (2005) is that a set of local learning rules result in synaptic updates that are on average equivalent to those produced by the error-backpropagation algorithm.

The work in this thesis builds on the idea in Roelfsema and van Ooyen (2005), which is that synaptic plasticity is gated by a combination of attention-gated feedback and a global prediction error signal:

Global prediction error As discussed in section 2.2, a problem with the BP algorithm is that unit-specific error signals need to be propagated throughout the network, while there is no known mechanism that could achieve this. However, in the RL context, we only need to communicate a scalar prediction error, and there is significant evidence that this signal is present in the brain by way of neuromodulatory substances such as dopamine.

Attention-gated feedback Output units are assumed to have feedback connections back to the hidden layer which allow information about the selected action to flow back into the network. Synaptic plasticity is constrained to those pathways that have both feedforward and feedback activation, which is where the term 'attention' originates.

The combination of these mechanisms yields learning rules that are powerful enough to learn non-linear transformations, using information that is locally available at the synapses.

The approach of AGREL differs from another popular class of methods called policy-gradient methods (e.g. Legenstein et al., 2009; Seung, 2003; Williams, 1992) where each unit (or synapse, in Seung (2003)) is a local agent that tries to increase the global reward. In this way, no knowledge about the structure of the network is needed. However, such methods do not scale well to large, structured, networks for the intuitive reason that the correlation between local behavior and global rewards weakens with increasing network size. In Roelfsema and van Ooyen (2005) it was shown that AGREL outperforms REINFORCE (Williams, 1992), and Urbanczik and Senn (2009) showed that feedback about the selected action (albeit using neuromodulators instead of feedback connections) significantly increases the (learning) performance of spiking neural networks.

[14] This is also known as an immediate reward setting.

There has been other work on biologically plausible implementations of error-backpropagation, most notably the generalized recirculation algorithm (GeneRec) in O'Reilly (1996). Like AGREL, GeneRec presumes the existence of (roughly symmetric) feedback weights, but there are important differences. One difference is mechanistic: GeneRec uses the neat insight that the computation of the error term for hidden units can be computed as a difference between two activation phases, the "minus" phase and the "plus" phase. In the plus phase, both input and output units are "clamped" and in the minus phase only the input is clamped. By combining these activations from the different phases, hidden units can compute their error without specialized "error" connections in the network. Another, more important, difference is that GeneRec requires that there is a teaching signal that provides the desired out is clamped at the output layer (see also section 2.2 above), which is an assumption that AGREL does not require. Another interesting idea is that of Zipser and Rumelhart (1993) who suggest that there may be a separate "error network" to communicate errors to earlier processing layers; the main problem with this idea is that such "error" networks have not been found in the brain.

The AGREL architecture and learning rules were developed in a 1-of-c classification setting, and the links to the RL framework were not worked out in detail. In this thesis, we used the core ideas of AGREL, but recast the problem in terms of action-values, instead of the action-probabilities of AGREL. In AGREL, it was further assumed that the expected reward for a given action could be expressed as the probability of selecting that action, which cannot be assumed to hold in general. This assumption is no longer required for the models presented in this thesis. We show how the idea of attention gated feedback learning can be extended to learning multi-dimensional outputs (chapter 3), and how it can be used in learning sequential tasks that may require working memory (chapters 4-6).

3 Multi Dimensional Attention Gated Learning

Abstract

How does the brain learn to map multi-dimensional sensory inputs to multi-dimensional motor outputs when it can only observe single rewards for the coordinated outputs of the whole network of neurons that make up the brain? We develop MQ-AGREL, a biologically plausible multi-layer neural network model for multi-dimensional reinforcement learning. We demonstrate that MQ-AGREL can learn non-linear mappings from inputs to multi-dimensional outputs by using only scalar reward feedback. We further show that in MQ-AGREL the changes in the connection weights follow the gradient that minimizes global prediction error, and that all information required for synaptic plasticity is locally present. This chapter is based on Rombouts, van Ooyen, et al. (2012).

3.1 Introduction

Imagine learning to play squash. High dimensional sensory inputs give rise to patterns of neuronal activations. These in turn yield a rich motor output, moving legs and arms to hit the ball with the racket. Which actions were useful? Which were not? While humans are able to learn such complex tasks, it is not clear how the brain solves these high dimensional learning problems. In neuronal terms, the problem is one of credit assignment: how should the efficacies of synapses in the brain change to make useful actions more probable?

Reinforcement Learning (RL; Sutton and Barto, 1998) offers a mathematical framework for learning to select optimal actions in Markov Decision Processes. For each possible state of the world an agent tries to predict the expected reward values (Q-values) of all possible actions. It then selects the best one with high probability. However, estimating all action-values quickly becomes intractable in high dimensional spaces, the well-known curse of dimensionality (Bellman, 1957).

A natural way to solve high-dimensional learning problems is to decompose them. Imagine a task that requires actions by both hands. Instead of estimating values for joint actions (a simultaneous action by left and right hand), as a naive application of RL suggests, we can estimate the values for atomic actions independently. The optimal joint action can then be produced by selecting locally optimal atomic actions. While a variety of modular reinforcement learning systems based on this intuition exist, e.g. Chang, Ho, and Kaelbling (2004), Ring, Schaul, and Schmidhuber (2011), and Rothkopf and Ballard (2010), ideas on how such approaches could be implemented in biologically plausible neural networks have been lacking.

Williams' REINFORCE algorithm for training neural networks (Williams, 1992) can be adapted to train neural networks with a modular structure. However, REINFORCE lacks a mechanism to solve the learning problem in an efficient and biologically plausible way. Roelfsema and van Ooyen developed a biologically plausible learning scheme for training multi-layer neural networks by RL, Attention-Gated Reinforcement Learning (AGREL; Roelfsema and van Ooyen, 2005). However, AGREL can only learn 1-of-n classification tasks, as it is constrained to select only single actions.

Here, we develop MQ-AGREL, a biologically plausible network model for modular reinforcement learning of multiple concurrent atomic actions. We build on the ideas of AGREL to arrive at a neural network scheme that can learn to select optimal simultaneous actions based on a scalar reinforcement signal. For each output dimension, the model has a separate output layer. Units in these output layers try to learn Q-values (Sutton and Barto, 1998) for their associated atomic actions. Each output layer subsequently has a separate stochastic Winner-Take-All competition (WTA) to select each atomic action. We show that the model learns to minimize prediction errors by gradient descent on a global prediction error, and that all information required for synaptic plasticity is local.

Plasticity in MQ-AGREL is determined by two factors, as in Roelfsema and van Ooyen (2005). The first is a globally available neuromodulatory signal that communicates a global prediction error. Such a signal could be implemented by a neuromodulator such as dopamine (Roelfsema, van Ooyen, and Watanabe, 2010; Schultz, Dayan, and Montague, 1997). The second factor is a set of feedback connections from output layers back to the hidden layer that gates plasticity based on the atomic actions that were selected.

We study classification tasks which require linear and non-linear mappings from inputs to multiple simultaneous outputs. All tasks are single

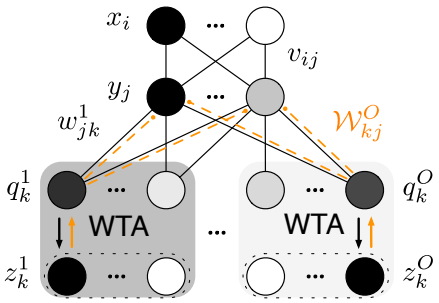


Figure 3.1: *MQ-AGREL architecture with O separate output layers (shading). Feedback weights shown with dashed lines.*

step input-output mappings where rewards are directly delivered, so that we only need to solve the spatial credit assignment problem. We show that MQ-AGREL can learn multi-dimensional non-linear tasks, and that it significantly outperforms the biologically implausible REINFORCE method (Williams, 1992).

3.2 MQ-AGREL

MQ-AGREL is modeled as a standard neural network with multiple Winner-Take-All (WTA) output layers o. In this way multiple atomic actions can be selected at the same time. The network predicts Q-values (Sutton and Barto, 1998) for all atomic actions. In each output layer, a stochastic WTA competition based on the predicted values selects one atomic action for execution. An example network is shown in Figure 3.1. For each joint action the network executes, the network receives a scalar reward r. The magnitude of this reward depends on the quality of the action (see section 3.3). The network learns by gradient descent on the global prediction error.

Input patterns x_i are presented to the input layer with N units. Hidden unit activations y_j are computed by a sigmoidal function of the linearly weighted summed input a_j:

$$y_j = \frac{1}{1+\exp(-a_j)} \quad \text{with} \quad a_j = \sum_{i=0}^{N} v_{ij} x_i \,, \tag{3.1}$$

where v_{ij} is the synaptic weight between input unit i and hidden unit j and v_{0j} denotes the bias weight. Each output layer o is fully connected to the hidden layer with M units by connections w^o_{jk}. Each output layer separately

computes Q-values q_k^o (Sutton and Barto, 1998):

$$q_k^o = \sum_{j=0}^{M} w_{jk}^o y_j \,, \tag{3.2}$$

where a bias unit y_0 is included.

With the Q-value estimates, a controller selects one of the atomic actions in each output layer. We implemented a max-Boltzmann controller (Wiering and Schmidhuber, 1997), which selects the action with the highest estimated Q-value with probability $1 - \epsilon$, and otherwise chooses an action with probabilities determined by the Boltzmann distribution:

$$Pr(z_k^o = 1) = \frac{\exp q_k^o}{\sum_{k'} \exp q_{k'}^o} \,. \tag{3.3}$$

The winning units K are then set to an activation of 1 and all other units to an activation of 0, $z_k^o = \delta_{kK}^o$ where δ_{kK} is the Kronecker delta function.

After executing an action the network receives a scalar reward r. A prediction error δ is computed as:

$$\delta = r - \sum_{o}^{O} \sum_{k}^{K^o} z_k^o q_k^o \,, \tag{3.4}$$

where O is the number of output layers and K^o denotes the number of output units in layer o. We assume that this δ signal is globally available.

Learning. The synaptic updates have two factors, as in AGREL (Roelfsema and van Ooyen, 2005). The first is the global prediction error δ and the second is a Hebbian interaction between feedforward activity and attentional feedback signals. The resulting learning rules are biologically plausible with all information required for the updates available at the synapses (Roelfsema and van Ooyen, 2005). The hidden to output layer synaptic updates are:

$$\Delta w_{jk}^o = \beta y_j z_k^o \delta \,, \tag{3.5}$$

where β is the learning rate. The synaptic updates between input and hidden layer are:

$$\Delta v_{ij} = \beta x_i y_j (1 - y_j) \delta \sum_{o}^{O} \sum_{k}^{K^o} \mathcal{W}_{kj}^o z_k^o \,, \tag{3.6}$$

where $\mathcal{W}^o_{kj}$ are feedback connections from the output layer to the hidden layer (Roelfsema and van Ooyen, 2005). These feedback connections effectively gate the plasticity of the higher level weights. For convenience, feedback and feedforward weights are assumed to be symmetrical; they can be trained by (3.5) as in Roelfsema and van Ooyen (2005).

MQ-AGREL and AGREL differ from (BP; Rumelhart, Hinton, and Williams, 1986) in two important respects. First, attention-gated learning does not require a teaching signal for each output node. Instead it is trained with a global prediction error. Second, it does not require the propagation of specific error signals from output layers to earlier layers (Roelfsema and van Ooyen, 2005; Roelfsema, van Ooyen, and Watanabe, 2010). These two elements make attention-gated learning schemes biologically plausible, unlike Error-Backpropagation.

MQ-AGREL minimizes global prediction error. The local learning rules (3.5)–(3.6) update the weights along the gradient on the global prediction error E:

$$E = \frac{1}{2}\left(r - \sum_o \sum_k z^o_k q^o_k\right)^2 = \frac{1}{2}\delta^2 \,. \tag{3.7}$$

We can derive the updates for weights between the hidden and output layer by moving in the opposite direction of the error gradient (Bishop, 1995):

$$-\frac{\partial E}{\partial w^o_{jk}} = \frac{\partial E}{\partial q^o_k}\frac{\partial q^o_k}{\partial w^o_{jk}} = \delta z^o_k y_j \,, \tag{3.8}$$

which is equivalent to the update in 3.5 up to the learning rate β. The updates for the input to hidden layer weights can be written:

$$-\frac{\partial E}{\partial v_{ij}} = \frac{\partial a_j}{\partial v_{ij}}\frac{\partial y_j}{\partial a_j}\sum_o^O \sum_k^{K^o} \frac{\partial E}{\partial q^o_k}\frac{\partial q^o_k}{\partial y_j} = x_i y_j (1-y_j)\delta \sum_o^O \sum_k^{K^o} \mathcal{W}^o_{kj} z^o_k \,, \tag{3.9}$$

where the rightmost term is the attentional feedback from the output layers via the feedback connections $\mathcal{W}^o_{kj}$. Because of the WTA competition, only one unit per output layer (with $z^o_k = 1$) contributes to the feedback. Again, the right hand side of (3.9) matches the update equation (3.6) up to a factor β.

This derivation shows that by combining attentional feedback signals and a globally available δ signal, MQ-AGREL minimizes prediction error by gradient descent on the Q-value prediction errors, using local updates.

Note that equations (3.4) and (3.7) implicitly assume that the rewards for the different modules are independent, but we will show that the network can deal to some extent with tasks where this assumption does not hold. Note that there are no guarantees that networks will learn the *correct* decomposition of the rewards, as all combinations of Q-values that equal the observed reward result in a prediction error of 0.

BP-REINFORCE. To compare MQ-AGREL with another learning algorithm, we implemented REINFORCE (Williams, 1992) for a network architecture with multiple WTA output layers. The architecture and activation functions are the same as in MQ-AGREL, except for the output layers, where actions were selected as in equation (3.3). We applied the REINFORCE updates for output weights (Williams, 1992):

$$\Delta w^o_{jk} = \beta r \Big[z^o_k - Pr(z^o_k = 1) \Big] y_j \,, \tag{3.10}$$

The hidden layer weights were updated by Error-Backpropagation (Rumelhart, Hinton, and Williams, 1986; Williams, 1992).

3.3 Experiments

Tasks. To demonstrate the performance of MQ-AGREL we implemented a set of binary tasks with increasing difficulty. The key aspects that we want to illuminate are that MQ-AGREL can deal with tasks that require non-linear transformations and that it scales well to multiple output layers. The tasks were constructed by concatenating different base tasks:

Linear The input consisted of two binary digits, of which one was randomly set to 1. The output was required to be the same as the input pattern. We provided the network with two hidden units for each linear task component.

XOR A version of the non-linear exclusive-OR problem. Two binary inputs need to be mapped to a 'match' signal if both inputs have the same value, and to a 'non-match' signal if they have different values. The output layer had two units, one coding for 'match' and the other for 'non-match'. We provided the network with two hidden units for each XOR task component.

Counting A set of N binary inputs is presented to the network. The network has to count the number of input units with activation 1 and

output this number in binary form. We provided an architecture with $\lceil \log_2 N \rceil + 1$ output layers, with two units each. We gave the network $\lceil \log_2 N \rceil + 2$ hidden units.

We evaluated the learning scheme on seven different tasks: Linear-Linear (LL), XOR-Linear (XL), XOR-XOR (XX), 3XOR (3X), 4XOR (4X) and the counting task with 8 (C8) and 32 (C32) inputs. On each trial the input for subtasks was selected independent of the input for the other subtasks. For instance, in the 3X task six random binary inputs were presented to the network, leading to a set of 2^6 possible input patterns. For each output layer the network received a reward of 1 if it was correct and 0 otherwise. The network only observed the total reward obtained.

Details on training. For all non-counting tasks, we trained networks for at most 250,000 random pattern presentations, or until convergence. Convergence was determined by keeping track of the rewards obtained in the last 200 trials. If the average amount of reward was at least 90% of the total possible reward, the network was said to have reached convergence. For the counting tasks, we set the maximal amount of training trials to 1,000,000. For all results reported here, we trained 100 networks with random initializations of the synaptic weights. Weights were sampled from a uniform distribution with range $[-0.25, 0.25]$. The exploration rate ϵ was set to 0.025. We computed the convergence rate (proportion of 100 networks that reached criterion) and 95% confidence intervals. For convergence times, we report Q1 (lower quartile), Q2 (median) and Q3 (upper quartile) plus minimal and maximal number of trials needed for convergence. Non-converged networks were assigned the maximal number of pattern presentations.

Performance. We compared MQ-AGREL to REINFORCE for the dual tasks (LL, XL and XX), as shown in Figure 3.2. While performance for the linear task is very comparable, MQ-AGREL significantly outperforms REINFORCE for tasks with a non-linear component. Substantial effort was made to find optimal parameters for both algorithms. We evaluated the algorithms with learning rates (β) of 0.01, 0.05, 0.10, 0.20, 0.30, 0.40; shown are the results for the best learning rate (highest convergence rate, fastest median convergence time) for both algorithms. For both algorithms 0.4 was the optimal rate for the LL and XL tasks. For the XX task the optimal rates were 0.30 for MQ-AGREL and 0.05 for REINFORCE. Due to the meagre

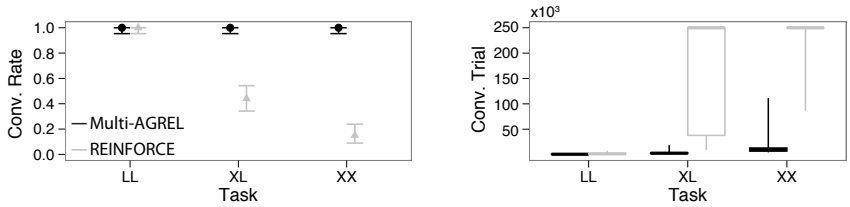


Figure 3.2: *Performance of MQ-AGREL (black) compared to REINFORCE (grey) on dual tasks, for best parameters found. Left, Convergence rates for 100 simulations in each condition. Error bars indicate* 95% *confidence intervals. Right, Box plots show (from bottom to top) minimal convergence trial, Q1 (lower quartile), Q2 (median, heavy line), Q3 (upper quartile) and maximal convergence trial. In REINFORCE, learning rates for each layer were individually optimized for best performance.*

performance of REINFORCE on these tasks, we do not show further results for this algorithm.

We further investigated the performance of MQ-AGREL on the harder 3X and 4X tasks (Figure 3.3a). Here MQ-AGREL also showed a robust performance. For the more difficult 4X task, the network worked best with a learning rate of 0.01. It might be that a lower learning rate smoothes out the noise on the learning signal that is caused by the other modules.

We also evaluated MQ-AGREL on the counting tasks (Figure 3.3b). These were significantly harder to learn than the 4X task (note the scale difference on the ordinate), but the algorithm managed to learn both tasks. There was no setting of β we tried in with all 100 networks managed to learn the C32 task, but it is likely that a longer maximal training time would improve the convergence rate. Note that the model had to train six disjoint WTA output layers in the C32 task, versus four in the C8 task.

As discussed in section 3.2, MQ-AGREL can also work in settings with dependent rewards. With dependent rewards, networks only receive rewards if all atomic actions are simultaneously correct, e.g. a reward of 2 if both sub-actions for a dual XOR task are correct, and 0 otherwise. Performance on the dual tasks (Figure 3.4a) is very comparable to that shown in Figure 3.2. The networks need significantly more trials to learn the 3X and 4X tasks with dependent rewards. Figure 3.4b shows the results for training MQ-AGREL for a maximum of 2.5×10^6 trials with a learning rate of 0.01. Note that tasks with dependent rewards have a degenerate nature: for each input pattern, only one of the 2^N binary action-vectors in the N-d space is rewarded and this quickly (with increasing N) makes it very hard to find the rewarded joint action. This problem is called the curse

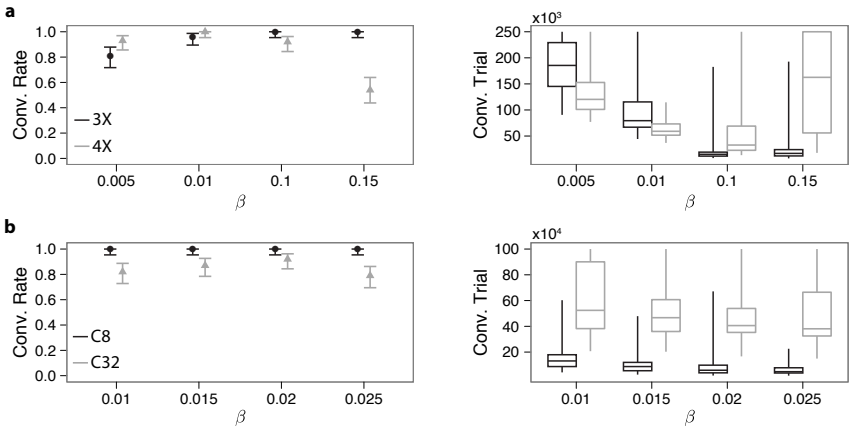


Figure 3.3: *Scaling performance, conventions as in Figure 3.2.* **a,** *Performance of MQ-AGREL on 3X (black) and 4X (grey) tasks.* **b,** *Performance of MQ-AGREL on C8 (black) and C32 (grey) tasks. Note that the learning rates (abscissa) and maximal training times (ordinate) differ from those in panel a.*

of dimensionality (Bellman, 1957), and MQ-AGREL cannot escape this. This curse can explain the sharp increase in the number of trials to learn the 4X task over the 3X task compared to the setting with independent rewards.

Finally, we tested MQ-AGREL on tasks where the rewards over the modules were not uniform. Figure 3.5 shows the results for networks that were trained on an XX task under four different reward ratios ($1:1, 2:1, 3:1, 4:1$, $\beta = 0.1$); the dots show the coordinates of winning Q-values pairs for all 16 different input patterns, over 100 networks, where we only show the results for correct trials. It is clearly visible that the networks are not guaranteed to find the true reward decomposition (circle), but that the summed Q-values tend to match the maximal possible reward (line). The centers of the distribution (crosses) are however biased toward the true reward decomposition. We conjecture that with a low enough learning rate and enough training trials, the estimated decomposition will get closer and closer to the true decomposition, because once the network finds the optimal action, exploratory sub-actions around the optimal joint action provide information for improving the reward estimates for individual modules. This reasoning also explains why the estimated reward decompositions in Figure 3.5 are biased towards the larger reward, as it induces a bigger learning signal when an exploratory sub-action for the large-reward module

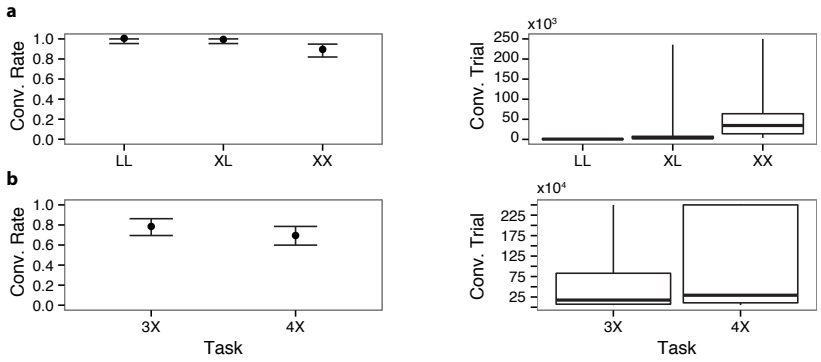


Figure 3.4: *Performance on tasks with dependent rewards, conventions as in Figure 3.2.* **a,** *Performance of MQ-AGREL on dual tasks (LL, XL, XX) with dependent rewards. Learning rates were* $\beta = 0.3, 0.2, 0.2$ *respectively, optimized over the same range as for the results in Figure 3.2.* **b,** *Performance of MQ-AGREL on 3X and 4X tasks with dependent rewards. Learning rate was set to* $\beta = 0.01$*. Note the difference in training times (ordinate) on the right plot.*

is selected.

3.4 Discussion

We have developed a biologically plausible neural network model that can learn a variety of difficult input-output mappings. Our work builds upon a previous model, AGREL Roelfsema and van Ooyen (2005), that could learn only single-dimensional input-output mappings, with 0, 1 rewards. Compared to AGREL, the controller in MQ-AGREL separates the action selection policy from the value estimation, allowing for the independent selection of atomic actions and the computation of a joint reward estimation. MQ-AGREL solves the spatial credit assignment problem by a combination of feedback signals and a globally released neuromodulatory signal which encodes a global prediction error. The feedback signals encode which atomic actions were selected, and constrain synaptic plasticity to those synapses that were involved in the selected joint action. We demonstrated that MQ-AGREL can learn difficult tasks, even when subtasks have a non-uniform distribution of rewards, or when rewards are not independent.

Compared to REINFORCE (Williams, 1992), MQ-AGREL exhibits much better convergence for tasks that contain non-linear components, even when REINFORCE uses the biologically implausible Error Backpropagation

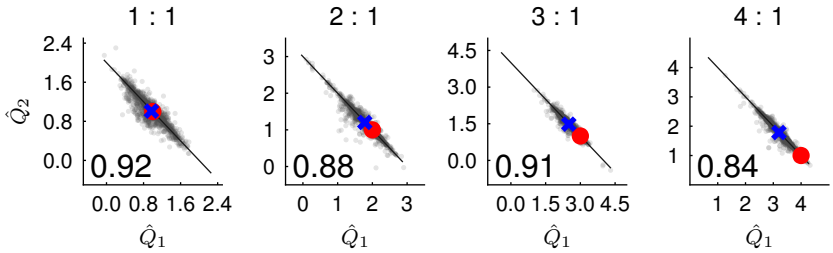


Figure 3.5: *Results for MQ-AGREL trained on XX task under unequal rewards (ratios on top). Lower left insets show proportion of* 100 *networks that learned the task. Small dots show coordinates for correctly winning Q-value combinations for all* 16 *input patterns for all networks. Black line is the line of equal global rewards. Red circle marks the true reward decomposition, blue cross marks mean estimated reward decomposition. We considered learning complete if the networks performed* > 90% *correct joint actions over a window of* 200 *trials.*

algorithm to update the hidden layer weights. A key difference is that REINFORCE updates the policies for all atomic actions after each decision, and not only those that were actually selected. Updating in this way may destroy the correct policy that was stored in the synapses for the non-executed atomic actions.

One of the limitations of MQ-AGREL is that networks are not guaranteed to learn the correct decomposition of rewards, even though they might learn the optimal policy. The method developed by Chang, Ho, and Kaelbling (2004) solves this issue by a Kalman filtering approach, but this is not a biologically plausible network solution. A Kalman filtering approach requires an internal model of the environment, which would fit with model-based RL, but this is beyond the scope of the current work.

A distinguishing factor in our model is the idea that multiple unrelated atomic actions can be active at the same time. Most other models assume that there is a single winning action, with all atomic actions mapping to the same output space (Ring, Schaul, and Schmidhuber, 2011; Rothkopf and Ballard, 2010). It is plausible that both types of solutions are used in the brain. Modules competing for the control of a single effector could use a shared action space model, and modules controlling independent effectors could use a model like MQ-AGREL.

Interestingly, recent experimental work has investigated whether humans could learn to simultaneously solve two independently rewarded tasks with two different hands (Gershman, Pesaran, and Daw, 2009). The

authors found that a reinforcement learning model that modularizes the two learning problems fits the behavioral data significantly better than one that learns action values over joint actions. This result gives experimental support for the idea that multiple simultaneous actions are indeed learned by separate modules.

MQ-AGREL is able to learn mappings for co-activated output modules based on a single globally available reward prediction error. This is not a trivial result, as it is not obvious that a global reward prediction error combined with local feedback is powerful enough to correctly solve the spatial credit assignment problem. As is mentioned in Chang, Ho, and Kaelbling (2004) and Gershman, Pesaran, and Daw (2009), module-specific prediction errors would be best for training separate output modules. However, the dopamine signal found in experiments seems to be unitary throughout the brain (Schultz, Dayan, and Montague, 1997).

Gershman, Pesaran, and Daw (2009) further report fMRI evidence that supports the idea that both value and learning related signals are lateralized. However, they used only two types of RL models as regressors: an RL model that learns values of joint actions (i.e. the naive RL approach) and a decomposed RL model that makes use of an effector-specific reward signal (i.e. a vector-valued prediction error). They did not consider an RL model that learns effector-specific action values based on a single reward prediction error, as in MQ-AGREL, and it would be interesting to compare such a model with the models considered in Gershman, Pesaran, and Daw (2009). Our model and simulations provide evidence that multiple modules could indeed be trained with a unitary reward prediction error.

4 Temporal Attention Gated Learning

Abstract

This chapter introduces the Attention-gated MEmory Tagging model, or AuGMEnT. This model extends the ideas of AGREL (Chapter 2) and MQ-AGREL (Chapter 3) to the delayed reward setting, resulting in a biologically plausible implementation of SARSA(λ) (Rummery and Niranjan, 1994). The basic model is limited to learning in settings where the optimal action can be inferred from the direct observation, an assumption that is often not met in real world tasks. Real world tasks almost always require some form of working memory, for instance for remembering what a road-sign said when arriving at an intersection. We therefore included active working memory in the model, based on the widespread presence of neurons with persistent activations throughout the brain. We show that AuGMEnT learns by stochastic gradient descent[1] on the temporal difference error. We will then show that AuGMEnT is indeed able to learn difficult tasks that require working memory and non-linear transformations, and that, when the model is trained on typical working memory tasks in the neuroscience literature, the model learns representations that are similar to those found in animals. This chapter is based on Rombouts, Bohte, and Roelfsema (2012) and Rombouts, Bohte, and Roelfsema (2015).

4.1 Introduction

Animals like monkeys can be trained to perform complex cognitive tasks, simply by giving rewards at the right times. They can learn to map sensory stimuli onto responses, to store task-relevant information and to integrate and combine unreliable sensory evidence. Training induces new stimulus and memory representations in 'multiple-demand' regions of the cortex (Duncan, 2010). For example, if monkeys are trained to memorize the

[1] Also known as online gradient descent.

location of a visual stimulus, neurons in lateral intra-parietal cortex (LIP) represent this location as a persistent increase of their firing rate (Gnadt and Andersen, 1988; Gottlieb and Goldberg, 1999). However, if the animals learn a visual categorization task, persistent activity of LIP cells becomes tuned to the boundary between categories (Freedman and Assad, 2006) whereas the neurons integrate probabilistic evidence if the task is sensory decision making (Yang and Shadlen, 2007). Similar effects of training on persistent activity have been observed in the somatosensory system. If monkeys are trained to compare frequencies of successive vibrotactile stimuli, working memory representations of analog variables are formed in somatosensory, prefrontal and motor cortex (Hernández et al., 1997).

Which learning mechanism induces appropriate working memories in these tasks? We here introduce AuGMEnT (Attention-Gated MEmory Tagging), a new reinforcement learning (Sutton and Barto, 1998) scheme that explains the formation of working memories during trial-and-error learning. AuGMEnT addresses two well-known problems in learning theory: temporal and structural credit-assignment (Rumelhart, Hinton, and Williams, 1986; Sutton and Barto, 1998). The temporal credit-assignment problem arises if an agent has to learn actions that are only rewarded after a sequence of intervening actions, so that it is difficult to assign credit to the appropriate ones. AuGMEnT solves this problem like previous temporal-difference reinforcement learning (RL) theories (Sutton and Barto, 1998). It learns action-values (known as *Q*-values, Sutton and Barto, 1998), i.e. the amount of reward that is predicted for a particular action when executed in a particular state of the world. If the outcome deviates from the reward-prediction, a neuromodulatory signal that codes the global reward-prediction error (RPE) gates synaptic plasticity in order to change the *Q*-value, in accordance with experimental findings (Dayan and Balleine, 2002; Montague, Hyman, and Cohen, 2004; Morris et al., 2006; Schultz, 2007). The key new property of AuGMEnT is that it can also learn tasks that require working memory, thus going beyond standard RL models (Sutton and Barto, 1998; Todd, Niv, and Cohen, 2009).

AuGMEnT also solves the structural credit-assignment problem of networks with multiple layers. Which synapses should change to improve performance? AuGMEnT solves this problem with an 'attentional' feedback mechanism. The output layer has feedback connections to units at earlier levels that provide feedback to those units that were responsible for the action that was selected (Roelfsema and van Ooyen, 2005). We propose that this feedback signal tags (Cassenaer and Laurent, 2012) relevant synapses and that the persistence of tags (known as eligibility traces, Houk, Adams,

and Barto, 1995; Sutton and Barto, 1998) permits learning if time passes between the action and the RPE. In this chapter we will introduce AuGMEnT, show that it learns by stochastic gradient descent on the temporal difference error, and discuss the relation to the SARSA(λ) (Rummery and Niranjan, 1994) algorithm. We will then use AuGMEnT to model a wide selection of working memory tasks from the literature and discuss its neuroscientific plausibility.

4.2 Model Architecture

We used AuGMEnT to train networks composed of three layers of units connected by two layers of modifiable synapses (Figure 4.1). Time was modeled in discrete steps.

Input layer

At the start of every time step, feedforward connections propagate information from the sensory layer to the association layer through modifiable connections v_{ij}. The sensory layer represents stimuli with instantaneous and transient units (Figure 4.1). Instantaneous units represent the current sensory stimulus $x(t)$ and are active as long as the stimulus is present. Transient units represent changes in the stimulus and behave like 'on (+)' and 'off (−)' cells in sensory cortices (Nassi and Callaway, 2009). They encode positive and negative changes in sensory inputs w.r.t. the previous time-step $t-1$:

$$x^{+}(t) = [x(t) - x(t-1)]_{+}\,, \tag{4.1}$$

$$x^{-}(t) = [x(t-1) - x(t)]_{+}\,, \tag{4.2}$$

where $[\cdot]_{+}$ is a threshold operation that returns 0 for all negative inputs, but leaves positive inputs unchanged. Every input is therefore represented by three sensory units. We assume that prior to trial start, i.e. at $t = 0$, all units have zero activation, and that the first time-step is $t = 1$.

Association layer

The second (hidden) layer of the network models the association cortex, and contains regular units (circles in Figure 4.1) and memory units (diamonds). We use the term 'regular unit' to reflect the fact that these are regular sigmoidal units that do not exhibit persistent activity in the absence of input. Regular units j are fully connected to instantaneous units i in the

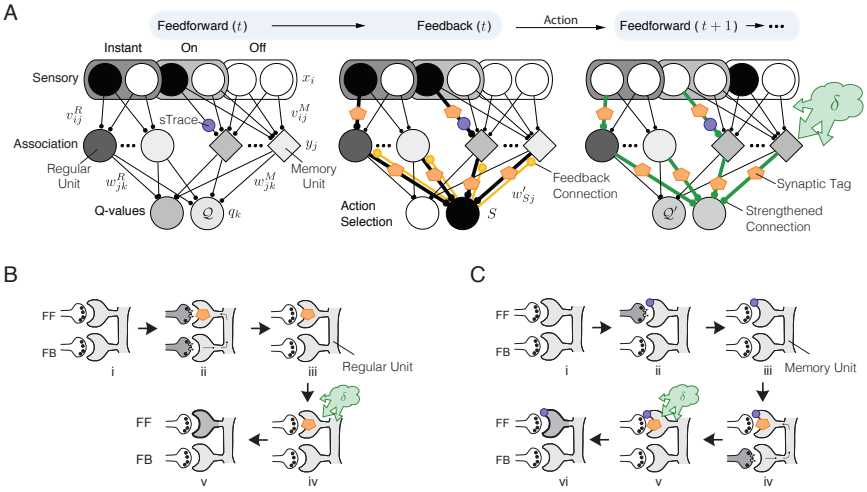


Figure 4.1: ***Model Architecture.*** **A**, *The model consists of a sensory input layer with units that code the input (instantaneous units) and transient units that only respond when a stimulus appears (on-units) or if it disappears (off-units). The association layer contains regular units (circles) with activities that depend on instantaneous input units, and integrating memory units (diamonds) that receive input from transient sensory units. The connections from the input layer to the memory cells maintain a synaptic trace (sTrace; blue circle) if the synapse was active. Units in the third layer code the value of actions (Q-values). After computing feed-forward activations, a Winner-Take-All competition determines the winning action (see middle panel). Action selection causes a feedback signal to earlier levels (through feedback connections w'_{Sj}, see middle panel) that lays down synaptic tags (orange pentagons) at synapses that are responsible for the selected action. If the predicted Q-value of the next action S' (Q'_S) plus the obtained reward $r(t)$ is higher than Q_S, a globally released neuromodulator δ (see eq. (4.17)) interacts with the tagged synapses to increase the strength of tagged synapses (green connections). If the predicted value is lower than expected, the strength of tagged synapses is decreased.* **B**, *Schematic illustration of the tagging process for regular units. FF is a feed-forward connection and FB is a feedback connection. The combination of feed-forward and feedback activation gives rise to a synaptic tag in step ii. Tags interact with the globally released neuromodulator δ to change the synaptic strength (step iv,v).* **C**, *Tagging process for memory units. Any presynaptic feed-forward activation gives rise to a synaptic trace (step ii; sTrace - purple circle). A feedback signal from the Q-value unit selected for action creates synaptic tags on synapses that carry a synaptic trace (step iv). The neuromodulator can interact with the tags to modify synaptic strength (v,vi).*

sensory layer by connections v_{ij}^R (the superscript R indexes synapses onto regular units, and v_{0j}^R is a bias weight). Their activity $y_j^R(t)$ is determined

by:

$$inp_j^R(t) = \sum_i v_{ij}^R x_i(t)\,, \tag{4.3}$$

$$y_j^R(t) = \sigma(inp_j^R(t))\,, \tag{4.4}$$

where $inp_j^R(t)$ denotes the synaptic input and σ a sigmoidal activation function;

$$\sigma(inp_j^R(t)) = \frac{1}{(1+\exp(\theta - inp_j^R(t)))}\,, \tag{4.5}$$

although our results do not depend on this particular choice of σ. The derivative of $y_j^R(t)$ can be conveniently expressed as:

$$y_j'^R(t) = \sigma'(inp_j^R(t)) = \frac{\partial y_j^R(t)}{\partial inp_j^R(t)} = y_j^R(t)(1-y_j^R(t))\,. \tag{4.6}$$

Memory units m (diamonds in Figure 4.1) are fully connected to the transient (+/−) units in the sensory layer by connections v_{lm}^M (superscript M indexes synapses onto memory units) and they integrate their input over the duration of the trial:

$$inp_m^M(t) = inp_m^M(t-1) + \sum_l v_{lm}^M x_l'(t)\,, \tag{4.7}$$

$$y_m^M(t) = \sigma(inp_m^M(t))\,, \tag{4.8}$$

where we use the shorthand x_l' that stands for both + and − cells, so $\sum_l v_{lm}^M x_l'(t)$ should be read as $\sum_l v_{lm}^{M+} x_l'^{+}(t) + \sum_l v_{lm}^{M-} x_l'^{-}(t)$. The selective connectivity between the transient input units and memory cells is advantageous. We found that the learning scheme is less stable when memory units also receive input from the instantaneous input units because even weak input becomes integrated across many time steps. We note, however, that there are other neuronal mechanisms which can prevent the integration of constant inputs. For example, the synapses between instantaneous input units and memory units could be rapidly adapting, so that only variations in input are integrated.

The simulated integration process causes persistent changes in the activity of memory units. It is easy to see that the activity of a memory unit equals the activity of a hypothetical regular unit that would receive input from all previous time-steps of the trial at the same time. To keep the model simple, we do not simulate the mechanisms responsible for

persistent activity, which have been addressed in previous work (Engel and Wang, 2011; Fransén et al., 2006; Koulakov et al., 2002). Although the perfect integration assumed in equation (4.7) does not exist in reality, we suggest that it is an acceptable approximation for trials with a relatively short duration as in the tasks that will be described in this chapter. Indeed, there are reports of single neuron integrators in entorhinal cortex with stable firing rates that persist for ten minutes or more (Egorov et al., 2002), which is orders of magnitude longer than the trials modeled in this chapter. In neurophysiological studies in behaving animals, the neurons that behave like regular and memory units in e.g. LIP (Gnadt and Andersen, 1988; Gottlieb and Goldberg, 1999) and frontal cortex (Funahashi, Bruce, and Goldman-Rakic, 1989) would be classified as visual cells and memory cells, respectively.

Q-value layer

The third layer receives input from the association layer through plastic connections w_{jk} (Figure 4.1). Its task is to compute action-values (i.e. Q-values, Sutton and Barto, 1998) for every possible action. Specifically, a Q-value unit aims to represent the (discounted) expected reward for the remainder of a trial if the network selects an action a in the current state s (Sutton and Barto, 1998):

$$Q^{\pi}(s,a) = E_{\pi}[R_t | s_t = s, a_t = a]\,, \qquad \text{with} \qquad R_t = \sum_{p=0}^{\infty} \gamma^p r_{t+p+1}\,, \tag{4.9}$$

where $E_{\pi}[\cdot]$ is the expected discounted future reward R_t given a and s, under action-selection policy π and $\gamma \in [0,1]$ determines the discounting of future rewards r. It is informative to explicitly write out the above expectation to see that Q-values are recursively defined as:

$$Q^{\pi}(s,a) = \sum_{s' \in S} P_{sa}^{s'} \left[R_{sa}^{s'} + \gamma \sum_{a' \in A} \pi(a', s') Q^{\pi}(s', a') \right], \tag{4.10}$$

where $P_{sa}^{s'}$ is a transition matrix, containing the probabilities that executing action a in state s will move the agent to state s', $R_{sa}^{s'}$ is the expected reward for this transition, and S and A are the sets of states and actions, respectively. Note that the action selection policy π is assumed to be stochastic in general. By executing the policy π, an agent samples trajectories according to the probability distributions π, $P_{sa}^{s'}$ and $R_{sa}^{s'}$ where every observed transition can

be used to update the original prediction $Q(s_t, a_t)$. Importantly, temporal difference learning schemes such as AuGMEnT are *model-free*, which means that they do not need explicit access to these probability distributions while improving their Q-values.

Q-value units k are fully connected to the association layer by connections w^R_{jk} (from regular units) and w^M_{mk} (from memory units). The action value $q_k(t)$ is estimated as:

$$q_k(t) = \sum_m w^M_{mk} y^M_m(t) + \sum_j w^R_{jk} y^R_j(t)\,, \tag{4.11}$$

where $q_k(t)$ aims to represent the value of action k at time step t, i.e. if $a_t = k$. In AuGMEnT, the state s in equation (4.9) is represented by the vector of activations in the association layer. Association layer units must therefore learn to represent and memorize information about the environment to compute the value of all possible actions a. They transform a so-called partially observable Markov decision process (POMDP) where the optimal decision depends on information presented in the past into a simpler Markov decision process (MDP) by storing relevant information as persistent activity, making it available for the next decision.

Action Selection

The action-selection policy π is implemented by a stochastic winner-takes-all (WTA) competition biased by the Q-values. The network usually chooses the action a with the highest value, but occasionally explores other actions to improve its value estimates. We used a Max-Boltzmann controller (Wiering and Schmidhuber, 1997) to implement the action selection policy π. It selects the greedy action (highest $q_k(t)$, ties are broken randomly) with probability $1 - \epsilon$, and a random action k sampled from the Boltzmann distribution P_B with small probability ϵ:

$$P_B(k) = \frac{\exp(q_k)}{\sum_{k'} \exp q'_k}\,. \tag{4.12}$$

This controller ensures that the model explores all actions, but usually selects the one with the highest expected value. We assume that the controller is implemented downstream, e.g. in the motor cortex or basal ganglia, but do not simulate the details of action selection, which have been addressed previously (Gurney, Prescott, and Redgrave, 2001; Humphries, Stewart, and Gurney, 2006; Stewart, Bekolay, and Eliasmith, 2012; Usher and McClelland, 2001). After selecting an action a, the activity

in the third layer becomes $z_k = \delta_{ka}$, where δ_{ka} is the Kronecker delta function (1 if $k = a$ and 0 otherwise). In other words, the selected action is the only one active after the selection process, and it then provides an "attentional" feedback signal to the association cortex (orange feedback connections in Figure 4.1A).

4.3 Learning

Learning in the network is controlled by two factors that gate plasticity: a global neuromodulatory signal and an attentional feedback signal. Once an action is selected, the unit that codes the winning action a feeds back to earlier processing levels to create synaptic tags (Frey and Morris, 1997; Moncada et al., 2011), also known as eligibility traces (Houk, Adams, and Barto, 1995; Sutton and Barto, 1998) on the responsible synapses (orange pentagons in Figure 4.1). Tagging of connections from the association layer to the motor layer follows a form of Hebbian plasticity: the tag strength depends on presynaptic activity (y_j) and postsynaptic activity *after* action selection (z_k) and tags thus only form at synapses w_{ja} onto the winning (i.e. selected) motor unit a,

$$\begin{aligned}
&\Delta Tag_{jk} = -\alpha Tag_{ja} + y_j z_k \,, && \text{which is equivalent to:} \\
&\Delta Tag_{ja} = -\alpha Tag_{ja} + y_j \,, && \text{for the winning action } a\text{, because } z_a = 1 \text{ and} \\
&\Delta Tag_{jk} = -\alpha Tag_{jk} \,, && \text{for } k \neq a, \quad \text{because} \quad z_{k \neq a} = 0 \,,
\end{aligned} \tag{4.13}$$

where α controls the decay of tags. Here, Δ denotes the change in one time-step, i.e. $Tag(t+1) = Tag(t) + \Delta Tag(t)$, and $Tag(t=0) = 0$. The formation of tags on the feedback connections w'_{aj} follows the same rule so that the strength of feedforward and feedback connections becomes similar during learning, in accordance with neurophysiological findings (Mao et al., 2011). Thus, the association units that provided strong input to the winning action a also receive strongest feedback (Figure 4.1, middle panel): they will be held responsible for the outcome of a. Importantly, the attentional feedback signal also guides the formation of tags on connections v_{ij} so that synapses from the input layer onto responsible association units j (strong w'_{aj}) are most strongly tagged (Figure 4.1B). For regular units we propose:

$$\Delta Tag_{ij} = -\alpha Tag_{ij} + x_i \sigma'(inp_j) w'_{aj} \,, \tag{4.14}$$

where σ' is the derivative of the association unit's activation function σ (equation (4.8)), which determines the influence that a change in the input

inp_j has on the activity of unit j. The idea has been illustrated in Figure 4.1B. Feedback from the winning action (lower synapse in Figure 4.1B) enables the formation of tags on the feedforward connections onto the regular unit. These tags can interact with globally released neuromodulators that inform all synapses about the RPE (green cloud 'δ' in Figure 4.1). Note that feedback connections only influence the plasticity of representations in the association layer but do not influence activity, in the present version of the model. We will come back to this point in the discussion.

In addition to synaptic tags, AuGMEnT uses synaptic traces (sTrace, blue circle in Figure 4.1A,C) for the learning of new working memories. These traces are located on the synapses from the sensory units onto memory cells. Any pre-synaptic activity in these synapses leaves a trace that persists for the duration of a trial. If one of the selected actions provides a feedback signal (panel iv in Figure 4.1C) to the post-synaptic memory unit, the trace gives rise to a tag making the synapse plastic as it can now interact with globally released neuromodulators:

$$\mathrm{d}sTrace_{ij} = x_i \quad \text{with} \quad (sTrace(t=0)=0)\,, \tag{4.15}$$

$$\Delta Tag_{ij} = -\alpha Tag_{ij} + sTrace_{ij}\sigma'(inp_j)w'_{aj}\,. \tag{4.16}$$

The traces persist for the duration of the trial, but all tags decay exponentially ($0 < \alpha < 1$). We assume that the time scale of trace updates is fast compared to the tag updates, indicated by d, so that tags are updated with the latest traces. After executing an action, the network may receive a reward $r(t)$. Moreover, an action a at time step $t-1$ may have caused a change in the sensory stimulus. For example, in most studies of monkey vision, a visual stimulus appears if the animal directs gaze to a fixation point. In the model, the new stimulus causes feedforward processing on the next time step t, which results in another set of Q-values. To evaluate whether a was better or worse than expected, the model compares the predicted outcome $Q_a(t-1)$, which has to be temporarily stored in the system, to the sum of the reward $r(t)$ and the discounted action-value $Q_{a'}(t)$ of unit a' that wins the subsequent stochastic WTA-competition. This temporal difference learning rule is known as SARSA (Rummery and Niranjan, 1994; Sutton and Barto, 1998):

$$\delta(t) = r(t) + \gamma q'_a(t) - q_a(t-1)\,. \tag{4.17}$$

The RPE $\delta(t)$ is positive if the outcome of a is better than expected and negative if it is worse. The representation of action values is biologically plausible because neurons coding action-value have been found in the frontal cor-

tex, basal ganglia and midbrain (Hikosaka, 2005; Morris et al., 2006; Samejima et al., 2005) and some orbitofrontal neurons specifically code the chosen value, q_a (Padoa-Schioppa and Assad, 2006). Moreover, dopamine neurons in the ventral tegmental area and substantia nigra represent δ (Montague, Hyman, and Cohen, 2004; Schultz, 2002, 2007). In the model, the release of neuromodulators makes δ available throughout the brain (green cloud in Figure 4.1). Plasticity of all synapses depends on the product of δ and tag strength:

$$\begin{aligned} \Delta v_{ij} &= \beta \delta(t) Tag_{ij} , \\ \Delta w_{jk} &= \beta \delta(t) Tag_{jk} , \end{aligned} \tag{4.18}$$

where β is the learning rate, and where the latter equation also holds for the feedback weights w'_{kj}. These equations capture the key idea of AuGMEnT: tagged synapses are held accountable for the RPE and change their strength accordingly. Note that AuGMEnT uses a four-factor learning rule for synapses v_{ij}. The first two factors are the pre- and postsynaptic activity that determine the formation of tags (equations (4.13), (4.14), (4.16)). The third factor is the "attentional" feedback from the motor selection stage, which ensures that tags are only formed in the circuit that is responsible for the selected action. The fourth factor is the RPE δ, which reflects whether the outcome of an action was better or worse than expected and determines if the tagged synapses increase or decrease in strength. Note that the computation of the RPE demands the comparison of Q-values in different time-steps. The RPE at time t depends on the action that the network selected at $t-1$ (see equation (4.17) and the next section), but the activity of the units that gave rise to this selection have typically changed at time t. The synaptic tags solve this problem because they labeled those synapses that were responsible for the selection of the previous action.

AuGMEnT is biologically plausible because the equations that govern the formation of synaptic tags (equations (4.13),(4.14),(4.16)) and traces (equation (4.15)) and the equations that govern plasticity (equation (4.18)) rely only on information that is available locally, at the synapse. Furthermore, the hypothesis that a neuromodulatory signal, like dopamine, broadcasts the RPE to all synapses in the network is supported by neurobiological findings (Montague, Hyman, and Cohen, 2004; Schultz, 2002, 2007).

In the next section we present the main theoretical result, which is that the AuGMEnT learning rules minimize the temporal difference errors

(equation (4.17)) of the transitions that are experienced by the network by stochastic gradient descent. Although AuGMEnT is not guaranteed to find optimal solutions (we cannot provide a proof of convergence), we found that it reliably learns difficult non-linear working memory problems, as will be illustrated below.

AuGMEnT minimizes the reward-prediction error (RPE)

The aim of AuGMEnT is to reduce the RPE $\delta(t)$ because low RPEs for all network states imply reliable Q-values so that the network can choose the action that maximizes reward at every time-step. The RPE $\delta(t)$ implies a comparison between two quantities: the *predicted* Q-value before the transition, $q_a(t-1)$, and a *target* Q-value $r(t)+\gamma q_{a'}(t)$, which consists of the actually observed reward and the next predicted Q-value (Sutton and Barto, 1998). If the two terms cancel, the prediction was correct. SARSA aims to minimize the prediction error by adjusting the networks weights w to improve the prediction $q_a(t-1)$ to bring it closer to the observed value $r(t)+\gamma q_{a'}(t)$. It is convenient to do this through on-line gradient descent on the squared prediction error $E(q_a(t-1)) = \frac{1}{2}\left([r(t)+\gamma q_{a'}(t)]-q_a(t-1)\right)^2$ with respect to parameters w (Rummery and Niranjan, 1994; Sutton and Barto, 1998):

$$\Delta w \propto -\frac{\partial E(q_a(t-1))}{\partial w} = -\frac{\partial E(q_a(t-1))}{\partial q_a(t-1)}\frac{\partial q_a(t-1)}{\partial w} = \delta(t)\frac{\partial q_a(t-1)}{\partial w}\,, \quad (4.19)$$

where $\frac{\partial q_a(t-1)}{\partial w}$ is the gradient of the predicted Q-value $q_a(t-1)$ with respect to parameters w. We have also used $\delta(t) = -\frac{\partial E(q_a(t-1))}{\partial q_a(t-1)}$, which follows from the definition of $E(q_a(t-1))$. Note that E is defined with regard to the sampled transition only so that the definition typically differs between successive transitions experienced by the network. For notational convenience we will abbreviate $E(q_a(t-1))$ to E_{q_a} in the remainder of this chapter. We will refer to the negative of equation (4.19) as "error gradient".

The RPE is high if the sum of the reward $r(t)$ and discounted $q_{a'}(t)$ deviates strongly from the prediction $q_a(t-1)$ on the previous time step. As in other SARSA methods, the updating of synaptic weights is only performed for the transitions that the network actually experiences. In other words, AuGMEnT is a so-called "on policy" learning method (Sutton and Barto, 1998).

We will first establish the equivalence of stochastic gradient descent defined in equation (4.19) and the AuGMEnT learning rule for the synaptic

weights $w^R_{jk}(t)$ from the regular units onto the Q-value units (Figure 4.1). According to equation (4.19), weights w^R_{ja} for the chosen action $k = a$ on time step $t-1$ should change as:

$$\Delta w^R_{ja} \propto \delta(t) \frac{\partial q_a(t-1)}{\partial w^R_{ja}(t-1)}\,, \tag{4.20}$$

leaving the other weights $k \neq a$ unchanged.

We will now show that AuGMEnT causes equivalent changes in synaptic strength. It follows from equation (4.11) that the influence of w^R_{ja} on $q_a(t-1)$ (i.e. $\frac{\partial q_a(t-1)}{\partial w_j a^R(t-1)}$ in equation (4.20)) equals $y^R_j(t-1)$, the activity of association unit j on the previous time step. This result allows us to rewrite (4.20) as:

$$\Delta w^R_{ja} \propto -\frac{\partial E_{q_a}}{\partial w^R_{ja}(t-1)} = \delta(t) \frac{\partial q_a(t-1)}{\partial w^R_{ja}(t-1)} = \delta(t) y^R_j(t-1)\,. \tag{4.21}$$

Recall from equation (4.13) that the tags on synapses onto the winning output unit a are updated according to $\Delta Tag_{ja} = -\alpha Tag_{ja} + y_j$ (orange pentagons in Figure 4.1). In the special case $\alpha = 1$, it follows that on time step t, $Tag_{ja}(t) = y^R_j(t-1)$ and that tags on synapses onto output units $k \neq a$ are 0. As a result,

$$\Delta w^R_{ja} \propto \delta(t) y^R_j(t-1) = \delta(t) Tag_{ja}(t); \tag{4.22}$$

$$= \delta(t) Tag_{jk}(t)\,, \tag{4.23}$$

for the synapses onto the selected action a, and the second, generalized, equation follows from the fact that $\frac{\partial q_a(t-1)}{\partial w^R_{jk}(t-1)} = 0$ for output units $k \neq a$ that were not selected and therefore do not contribute to the RPE. Inspection of equations (4.18) and (4.23) reveals that AuGMEnT indeed takes a step of size β in the direction opposite to the error gradient of equation (4.19) (provided $\alpha = 1$; we discuss the case $\alpha \neq 1$ below). The updates for synapses between memory units m and Q-value units k are equivalent to those between regular units and the Q-value units. Thus,

$$\Delta w^M_{mk} \propto -\frac{\partial E_{q_a}}{\partial w^M_{mk}(t-1)} = \delta(t) \frac{\partial q_k(t-1)}{\partial w^M_{mk}(t-1)} = \delta(t) Tag_{mk}(t)\,. \tag{4.24}$$

The plasticity of the feedback connections w'^R_{kj} and w'^M_{km} from the Q-value layer to the association layer follows the same rule as the updates of connections w^R_{jk} and w^M_{mk} and the feedforward and feedback connections between two units therefore become proportional during learning (Roelfsema

and van Ooyen, 2005). We will now show that synapses v_{ij}^R between the input layer and the regular memory units (Figure 4.1) also change according to the negative gradient of the error function defined above. Applying the chain rule to compute the influence of v_{ij}^R on the error $E(t)$ results in the following equation:

$$\begin{aligned}\Delta v_{ij}^R \propto -\frac{\partial E_{q_a}}{\partial v_{ij}^R(t-1)} &= \delta(t)\frac{\partial q_a(t-1)}{\partial y_j^R(t-1)}\frac{\partial y_j^R(t-1)}{\partial inp_j^R(t-1)}\frac{\partial inp_j^R(t-1)}{\partial v_{ij}^R(t-1)}\,, \\ &= \delta(t) w_{ja}^R \sigma'(inp_j^R(t-1)) x_i(t-1)\,. \end{aligned} \tag{4.25}$$

The amount of attentional feedback that was received by unit j from the selected Q-value unit a at time $t-1$ is equal to w_{aj}^R because the activity of unit a equals 1 once it has been selected. As indicated above, learning makes the strength of feedforward and feedback connections similar so that w_{ja}^R can be estimated as the amount of feedback $w_{aj}'^R$ that unit j receives from the selected action a,

$$-\frac{\partial E_{q_a}}{\partial v_{ij}^R(t-1)} = \delta(t) w_{aj}'^R \sigma'(inp_j^R(t-1)) x_i(t-1)\,, \tag{4.26}$$

Recall from equation (4.14) that the tags on synapses v_{ij}^R are updated according to $\Delta Tag_{ij} = -\alpha Tag_{ij} + x_i \sigma'(inp_j) w_{aj}'^R$. Figure 4.1B illustrates how feedback from action a controls the tag formation process. If $\alpha = 1$, then on time step t, $Tag_{ij}(t) = x_i(t-1)\sigma'(inp_j^R(t-1)) w_{aj}'^R$ so that equation (4.26) can be written as:

$$-\frac{\partial E_{q_a}}{\partial v_{ij}^R(t-1)} = \delta(t) Tag_{ij}(t)\,. \tag{4.27}$$

A comparison to equation (4.18) demonstrates that AuGMEnT also takes a step of size β in the direction opposite to the error gradient for these synapses.

The final set of synapses that needs to be considered are between the transient sensory units and the memory units. We approximate the total input $inp_m^M(t)$ of memory unit m as (see equation (4.7)):

$$\begin{aligned} inp_m^M(t) &= \sum_l v_{lm}^M(t) x_l'(t) + \sum_{l,t'=0}^{t-1} v_{lm}^M(t') x_l'(t')\,, \\ &\approx \sum_l v_{lm}^M(t) \sum_{t'=0}^{t} x_l'(t')\,, \end{aligned} \tag{4.28}$$

The approximation is good if synapses v^M_{lm} change slowly during a trial. According to equation (4.19), the update for these synapses is:

$$\Delta v^M_{lm} \propto -\frac{\partial E_{q_a}}{\partial v^M_{lm}} = \delta(t)\frac{\partial q_a(t-1)}{\partial y^M_m(t-1)}\frac{\partial y^M_m(t-1)}{\partial inp^M_m(t-1)}\frac{\partial inp^M_m(t-1)}{\partial v^M_{lm}(t-1)}\,,$$
$$= \delta(t) w'^M_{am}\sigma'(inp^M_m(t-1))[\sum_{t'=0}^{t-1} x'_l(t')]\,. \tag{4.29}$$

Equation (4.15) specifies that $\mathrm{d}sTrace_{lm} = x_l$ so that $sTrace_{lm} = \sum_{t'=0}^{t-1} x'_l(t')$, the total presynaptic activity of the input unit up to time $t-1$ (blue circle in Figure 4.1C). Thus, equation (4.29) can also be written as:

$$\Delta v^M_{lm} = \delta(t) w'^M_{am}\sigma'(y^M_m(t-1))sTrace_{lm}\,. \tag{4.30}$$

Equation (4.16) states that $\Delta Tag_{lm} = -\alpha Tag_{lm} + sTrace_{lm}\sigma'(inp^M_m)w'^M_{am}$, because the feedback from the winning action a converts the trace into a tag (panel iv in Figure 4.1C). Thus, if $\alpha = 1$ then $Tag^M_{lm}(t) = w'^M_{am}\sigma'(inp^M_m(t-1))sTrace_{lm}$ so that:

$$\Delta v^M_{lm} = \delta(t) Tag^M_{lm}(t)\,. \tag{4.31}$$

Again, a comparison of equations (4.31) and (4.18) shows that AuGMEnT takes a step of size β in the direction opposite to the error gradient, just as is the case for all other categories of synapses. We conclude that AuGMEnT causes a stochastic gradient descent of all synaptic weights to minimize the temporal difference error if $\alpha = 1$. As an aside, it is straightforward to generalize the learning rules for memory units to a case with imperfect, leaky, memory (with decay rate possibly varying as a function of inputs), see the appendix.

AuGMEnT and SARSA(λ)

AuGMEnT provides a biological implementation of the well known RL method called SARSA, although it also goes beyond traditional SARSA (Rummery and Niranjan, 1994; Sutton and Barto, 1998) by (i) including memory units (ii) representing the current state of the external world as a vector of activity at the input layer (iii) providing an association layer that aids in computing Q-values that depend non-linearly on the input, thus providing a biologically plausible equivalent of the error-backpropagation learning rule (Rumelhart, Hinton, and Williams, 1986), and (iv) using synaptic tags and traces (Figure 4.1B,C) so that all the information necessary for plasticity is available locally at every synapse.

The tags and traces determine the plasticity of memory units and aid in decreasing the RPE by improving the Q-value estimates. If a memory unit j receives input from input unit i then a trace of this input is maintained at synapse v_{ij} for the remainder of the trial (blue circle in Figure 4.1C). Suppose that j, in turn, is connected to action a which is selected at a later time point. Now unit j receives feedback from a so that the trace on synapse v_{ij} becomes a tag making it sensitive to the globally released neuromodulator that codes the RPE δ (panel iv in Figure 4.1C). If the outcome of a was better than expected ($\delta > 0$) (green cloud in panel v), v_{ij} strengthens (thicker synapse in panel vi). When the stimulus that activated unit i reappears on a later trial, the larger v_{ij} increases unit j's persistent activity which, in turn, enhances the activity of the Q-value unit representing a, thereby decreasing the RPE.

The synaptic tags of AuGMEnT correspond to the eligibility traces used in RL schemes such as SARSA(λ) (Rummery and Niranjan, 1994). In SARSA learning speeds up if the eligibility traces do not fully decay on every time step, but exponentially with parameter $\lambda \in [0, 1]$ (Sutton and Barto, 1998); the resulting rule is called SARSA(λ). In AuGMEnT, the parameter α plays an equivalent role and precise equivalence can be obtained by setting $\alpha = 1 - \lambda\gamma$ as can be verified by making this substitution in equations (4.13), (4.14) and (4.16) (noting that $Tag(t+1) = Tag(t) + \Delta Tag(t)$). It follows that tags decay exponentially as $Tag(t+1) = \lambda\gamma Tag(t)$, equivalent to the decay of eligibility traces in SARSA(λ). These results thereby establish the correspondence between the biologically inspired AuGMEnT learning scheme and the RL method SARSA(λ). A special condition occurs at the end of a trial. The activity of memory units, traces, tags, and Q-values are set to zero (see Sutton and Barto, 1998), after updating of the weights with a δ that reflects the transition to the terminal state.

In the next section we will illustrate how AuGMEnT can train multilayered networks with the form shown in Figure 4.1 to perform a large variety of tasks that have been used to study neuronal representations in the association cortex of monkeys.

4.4 Simulation of animal learning experiments

We tested AuGMEnT on four different tasks that have been used to investigate the learning of working memory representations in monkeys. The first three tasks have been used to study the influence of learning on neuronal activity in area LIP and the fourth task to study vibrotactile working mem-

ory in multiple cortical regions. All tasks have a similar overall structure: the monkey starts a trial by directing gaze to a fixation point or by touching a response key. Then stimuli are presented to the monkey and it has to respond with the correct action after a memory delay. At the end of a trial, the model could choose between two possible actions. The full task reward (r_f, 1.5 units) was given if this choice was correct, while we aborted trials and gave no reward if the model made the wrong choice or broke fixation (released the key) before a go signal.

Researchers usually train monkeys on these tasks with a shaping strategy. The monkey starts with simple tasks and then the complexity is gradually increased. It is also common to give small rewards for reaching intermediate goals in the task, such as attaining fixation. We encouraged fixation (or touching the key in the vibrotactile task below) by giving a small shaping reward (r_i, 0.2 units) if the model directed gaze to the fixation point (touched the key). In the next sections we will demonstrate that the training of networks with AuGMEnT is facilitated by shaping. Shaping was not necessary for learning in any of the tasks, however, but it enhanced learning speed and increased the proportion of networks that learned the task within the allotted number of training trials.

Across all the simulations, we used a single, fixed configuration of the association layer (three regular units, four memory units) and Q-layer (three units) and a single set of learning parameters (Tables 4.1,4.2). The number of input units varied across tasks as the complexity of the sensory stimuli differed. We note, however, that the results described below would have been identical had we simulated a fixed, large input layer with silent input units in some of the tasks, because silent input units have no influence on activity in the rest of the network.

Saccade/Antisaccade Task

The first task (Figure 4.2A) is a memory saccade/anti-saccade task modeled after Gottlieb and Goldberg (1999). Every trial started with an empty screen, shown for one time step. Then a fixation mark was shown that was either black or white, indicating that a pro- or anti-saccade would be required. The model had to fixate within 10 time-steps, otherwise the trial was terminated without reward. If the model fixated for two time-steps, we presented a cue on the left or the right side of the screen for one time-step and gave the fixation reward r_i. This was followed by a memory delay of two time steps during which only the fixation point was visible. At the end of the memory delay the fixation mark turned off. To collect the final reward r_f in

the pro-saccade condition, the model had to make an eye-movement to the remembered location of the cue and to the opposite location on anti-saccade trials. The trial was aborted if the model failed to respond within eight time steps.

The input units of the model (Figure 4.2B) represented the color of the fixation point and the presence of the peripheral cues. The three Q-value units had to represent the value of directing gaze to the center, left and right side of the screen. This task can only be solved by storing cue location in working memory and, in addition, requires a non-linear transformation and can therefore not be solved by a linear mapping from the sensory units to the Q-value units. We trained the models for maximally $25,000$ trials, or until they learned the task. We kept track of accuracy for all four trial types as the proportion correct responses in the last 50 trials. When all accuracies reached 0.9 or higher, learning and exploration were disabled (i.e. β and ϵ were set to zero) and we considered learning successful if the model performed all trial-types accurately.

We found that learning of this task with AuGMEnT was efficient. We distinguished three points along the task learning trajectory: learning to obtain the fixation reward ('Fix'), learning to fixate until fixation-mark offset ('Go') and finally to correctly solve the task ('Task'). To determine the 'Fix'-learn trial, we determined the time point when the model attained fixation in 90 out of 100 consecutive trials. The model learned to fixate after 224 trials (median) (Figure 4.2C). The model learned to maintain gaze until the go signal after $\sim 1,300$ trials and it successfully learned the complete task after $\sim 4,100$ trials. Thus, the learning process was at least an order of magnitude faster than in monkeys that typically learn such a task after months of training with more than $1,000$ trials per day. To investigate the effect of the shaping strategy, we also trained $10,000$ networks without the extra fixation reward (r_i was zero). Networks that received fixation rewards were more likely to learn than networks that did not (99.5% versus 76.4%; $\chi^2 = 2,498$, $p < 10^{-6}$). Thus, shaping strategies facilitate training with AuGMEnT, similar to their beneficial effect in animal learning (Krueger and Dayan, 2009).

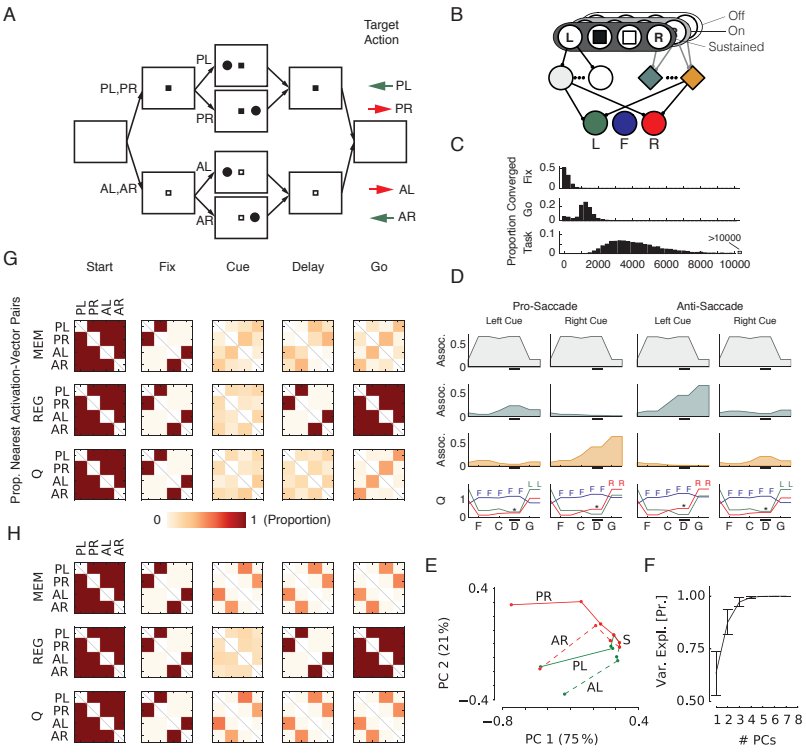

A
Target
Action
PL,PR
PL
PR
AL,AR
AL
AR
PL
PR
AL
AR
B
Off
On
Sustained
L
R
L F R
C
Proportion Converged
Fix
Go
Task
0.5
0.2
0.1
0
>10000
0 2000 4000 6000 8000 10000
G
Start
Fix
Cue
Delay
Go
Prop. Nearest Activation-Vector Pairs
MEM
REG
Q
PL
PR
AL
AR
0
1 (Proportion)
D
Pro-Saccade
Anti-Saccade
Left Cue
Right Cue
Assoc.
Q
F C D G
H
E
PC 2 (21 %)
PC 1 (75 %)
PR
AR
PL
AL
S
0.4
−0.4
−0.8
F
Var. Expl. [Pr.]
PCs
1.00
0.75
0.50
1 2 3 4 5 6 7 8

Figure 4.2

Figure 4.2 ***(continued from previous page): Saccade/antisaccade task.*** **A**, *Structure of the task, all possible trials have been illustrated. Fixation mark color indicates whether a saccade (P) or anti-saccade (A) is required after a memory delay. Colored arrows show the required action for the indicated trial types. L: cue left; R: cue right.* **B**, *The sensory layer represents the visual information (fixation point, cue left/right) with sustained and transient (on/off) units. Units in the Q-value layer code three possible eye positions: left (green), center (blue) and right (red).* **C**, *Time course of learning:* 10,000 *networks were trained, of which* 9,945 *learned the task within* 25,000 *trials. Histograms show the distribution of trials when the model learned to fixate ('fix'), maintain fixation until the 'go'-signal ('go') and learned the complete task ('task').* **D**, *Activity of example units in the association and Q-layer. The grey trace illustrates a regular unit and the green and orange traces memory units. The bottom graphs show activity of the Q-value layer cells. Colored letters denote the action with highest Q-value. Like the memory cells, Q-value units also have delay activity that is sensitive to cue location (∗ in the lower panel) and their activity increases after the go-signal.* **E**, *2D-PCA projection of sequence of association layer activations for the four different trial types for an example network. S marks the start of the trials (empty screen). Pro saccade trials are shown with solid lines and anti-saccade trials with dashed lines. Color indicates cue location (green - left; red - right) and labels indicate trial type (P/A = type pro/anti; L/R = cue left/right). Percentages on the axes show variance explained by the PCs.* **F**, *Mean variance explained as a function of the number of PCs over all* 100 *trained networks, error bars s.d.* **G**, *Pairwise analysis of activation vectors of different unit types in the network (see main text for explanation). MEM: memory; REG: regular. This panel is aligned with the events in panel (A). Each square within a matrix indicates the proportion of networks where the activity vectors of different trial types were most similar. Color scale is shown below. For example, the right top square for the memory unit matrix in the 'go' phase of the task indicates that around* 25% *of the networks had memory activation vectors that were most similar for Pro-Left and Anti-Right trials.* **H**, *Pairwise analysis of activation-vectors for networks trained on a version of the task where only pro-saccades were required. Conventions as in (G).*

The activity of a fully trained network is illustrated in Figure 2D. One of the association units (grey in Figure 4.2D) and the *Q*-unit for fixating at the center of the display (blue in Figure 4.2B,D) had strongest activity at fixation onset and throughout the fixation and memory delays. If recorded in a macaque monkey, these neurons would be classified as fixation cells. After the go-signal the *Q*-unit for the appropriate eye movement became more active. The activity of the *Q*-units also depended on cue-location during the memory delay as is observed, for example, in the frontal eye fields (∗ in Figure 4.2D) (Sommer and Wurtz, 2001). This activity is caused by the input from memory units in the association layer that memorized cue location as a persistent increase in their activity (green and orange in

Figure 4.2D). Memory units were also tuned to the color of the fixation mark which differentiated pro-saccade trials from anti-saccade trials, a conjoined selectivity necessary to solve this non-linear task (Rigotti et al., 2013). There was an interesting division of labor between regular and memory units in the association layer. Memory units learned to remember the cue location. In contrast, regular units learned to encode the presence of task-relevant sensory information on the screen. Specifically, the fixation unit in Fig. 4.2D (upper row) was active as long as the fixation point was present and switched off when it disappeared, thus cueing the model to make an eye movement. Interestingly, these two classes of memory neurons and regular ("light sensitive") neurons are also found in areas of the parietal and frontal cortex of monkeys (Gnadt and Andersen, 1988; Sommer and Wurtz, 2001) where they appear to have equivalent roles.

Figure 4.2D provides a first, casual impression of the representations that the network learns. To gain a deeper understanding of the representation in the association layer that supports the non-linear mapping from the sensory units to the Q-value units, we performed a principal component analysis (PCA) on the activations of the association units. We constructed a single (32×7) observation matrix from the association layer activations for each time-step (there were seven association units and eight time-points in each of the four trial-types), with the learning rate β and exploration rate ϵ of the network set to zero. Figure 4.2E shows the projection of the activation vectors onto the first two principal components for an example network. It can be seen activity in the association layer reflects the important events in the task. The color of the fixation point and the cue location provide information about the correct action and lead to a 'split' in the 2D principal component (PC) space. In the 'Go' phase, there are only two possible correct actions: 'left' for the Pro-Left and Anti-Right trials and 'right' otherwise. The 2D PC plot shows that the network splits the space into three parts based on the optimal action: here the 'left' action is clustered in the middle, and the two trial types with target action 'right' are adjacent to this cluster. This pattern (or its inversion with the 'right' action in the middle) was typical for the trained networks. Figure 2F shows how the explained variance in the activity of association units increases with the number of PCs, averaged over 100 simulated networks; most variance was captured by the first two PCs.

To investigate the representation that formed during learning across all simulated networks, we next evaluated the similarity of activation patterns (Euclidean distance) across the four trial types for the regular and memory association units and also for the units in the Q-value layer (Figure 4.2G).

For every network we entered a '1' in the matrix for trial types with the smallest distance and a '0' for all other pairs of trials and then aggregated results over all networks by averaging the resulting matrices. Initially the patterns of activity in the association layer are similar for all trial types, but they diverge after the presentation of the fixation point and the cue. The regular units convey a strong representation of the color of the fixation point (e.g. activity in pro-saccade trials with a left cue is similar to activity in pro-saccade trials with a right cue; PL and PR in Figure 2G), which is visible at all times. Memory units have a clear representation of the previous cue location during the delay (e.g. AL trials similar to PL trials and AR to PR trials in Figure 4.2G). At the go-cue their activity became similar for trials requiring the same action (e.g. AL trials became similar to PR trials), and the same was true for the units in the *Q*-value layer.

In our final experiment with this task, we investigated if working memories are formed specifically for task-relevant features. We used the same stimuli, but we now only required pro-saccades so that the color of the fixation point became irrelevant. We trained 100 networks, of which 96 learned the task and we investigated the similarities of the activation patterns. In these networks, the memory units became tuned to cue-location but not to color of the fixation point (Figure 4.2H; note the similar activity patterns for trials with a differently colored fixation point, e.g. AL and PL trials). Thus, AuGMEnT specifically induces selectivity for task-relevant features in the association layer.

Delayed Match-to-category Task

The selectivity of neurons in the association cortex of monkeys changes if the animals are trained to distinguish between categories of stimuli. After training, neurons in frontal (Freedman, Riesenhuber, et al., 2001) and parietal cortex (Freedman and Assad, 2006) respond similarly to stimuli from the same category and discriminate between stimuli from different categories. In one study (Freedman and Assad, 2006), monkeys had to group motion stimuli in two categories in a delayed-match-to-category task (Figure 4.3A). They first had to look at a fixation point, then a motion stimulus appeared and after a delay a second motion stimulus was presented. The monkeys' response depended on whether the two stimuli came from the same category or from different categories.

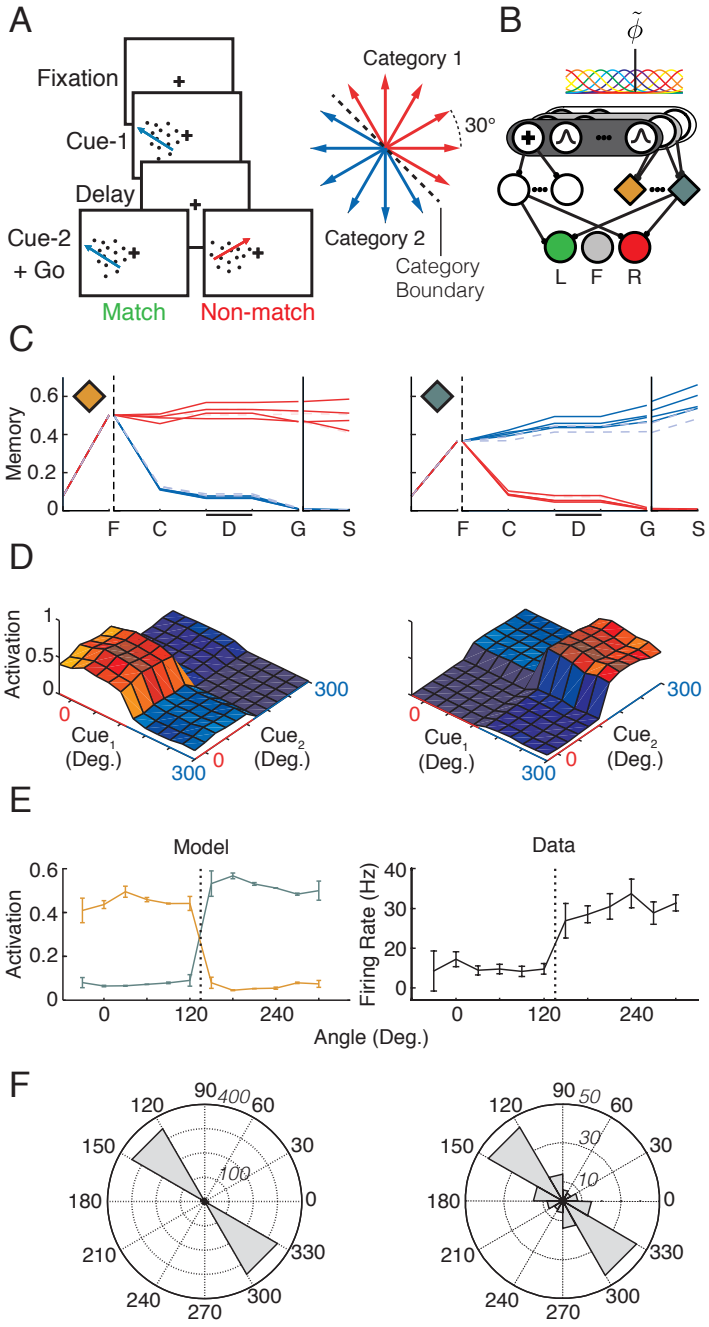

A
Fixation
Cue-1
Delay
Cue-2
+ Go
Match
Non-match
Category 1
30°
Category 2
Category
Boundary
B
L F R
C
Memory
0.6
0.4
0.2
0
F C D G S
D
Activation
1
0.5
0
Cue1
(Deg.)
Cue2
(Deg.)
0
300
E
Model
Data
Activation
Firing Rate (Hz)
Angle (Deg.)
F
90
120
60
150
30
180
0
210
330
240
270
300
400
100
50
30
10

Figure 4.3

Figure 4.3 ***(continued from previous page)*: *Match-to-category task.*** **A**, *When the network directed gaze to the fixation point, we presented a motion stimulus (cue-1), and after a delay a second motion stimulus (cue-2). The network had to make a saccade to the left when the two stimuli belonged to the same category (match) and to the right otherwise. There were twelve motion directions, which were divided into two categories (right).* **B**, *The sensory layer had a unit representing the fixation point and* 20 *units with circular Gaussian tuning curves (s.d.* 12 *deg.) with preferred directions evenly distributed over the unit circle. C, Activity of two example memory units in a trained network evoked by the twelve cue-1 directions. Each line represents one trial, and color represents cue category. Responses to cues closest to the categorization boundary are drawn with a dashed line of lighter color. F, fixation mark onset;* **C**, *cue-1 presentation. D, delay; G, cue-2 presentation (go signal); S, saccade.* **D**, *Activity of the same two example memory units as in (C) in the 'go' phase of the task for all* 12×12 *combinations of cues. Colors of labels and axes indicate cue category.* **E**, *Left, Motion tuning of the memory units (in C) at the end of the memory delay. Error bars show s.d. across trials and the dotted vertical line indicates the category boundary. Right, Tuning of a typical LIP neuron (from Freedman and Assad, 2006), error bars show s.e.m.* **F**, *Left, Distribution of the direction change that evoked the largest difference in response across memory units from 100 networks. Right, Distribution of direction changes that evoked largest response differences in LIP neurons (from Freedman and Assad, 2006).*

We investigated if AuGMEnT could train a network with an identical architecture (with 3 regular and 4 memory units in the association layer) as the network of the delayed saccade/antisaccade task to perform this categorization task. We used an input layer with a unit for the fixation point and 20 units with circular Gaussian tuning curves of the form:

$$r(x) = \exp\left(-\frac{(x-\theta_c)^2}{2\sigma^2}\right), \tag{4.32}$$

with preferred directions θ_c evenly distributed over the unit circle and a standard deviation σ of 12 deg. (Figure 4.3B). The two categories were defined by a boundary that separated the twelve motion directions (adjacent motion directions were separated by 30 deg.) into two sets of six directions each.

We first waited until the model directed gaze to the fixation point. Two time-steps after fixation we presented one of twelve motion-cues as cue-1 for one time step and gave the fixation reward r_i (Figure 4.3A). We added Gaussian noise to the motion direction (s.d. 5 deg.) to simulate noise in the sensory system. The model had to maintain fixation during the ensuing memory delay that lasted two time steps. We then presented a second

motion stimulus as cue-2 and the model had to make an eye-movement (either left or right; the fixation mark did not turn off in this task) that depended on the match between the categories of the cues. We required an eye movement to the left if both stimuli belonged to the same category and to the right otherwise, within eight time-steps after cue-2. We trained 100 models and measured accuracy for the preceding 50 trials with the same cue-1. We determined the duration of the learning phase as the trial where accuracy had reached 80% for all cue-1 types.

In spite of their simple feedforward structure with only seven units in the association layer, AuGMEnT trained the networks to criterion in all simulations within a median of 11,550 trials. Figure 4.3C illustrates motion tuning of two example memory neurons in a trained network. Both units had become category selective, from cue onset onwards and throughout the delay period. Figure 4.3D shows the activity of these units at 'Go' time (i.e. after presentation of cue-2) for all 144 combinations of the two cues. Figure 4.3E shows the tuning of the memory units during the delay period. For every memory unit of the simulations ($N = 400$), we determined the direction change eliciting the largest difference in activity (Figure 4.3F) and found that the units exhibited the largest changes in activity for differences in the motion direction that crossed a category boundary, as do neurons in LIP (Freedman and Assad, 2006) (Figure 4.3E,F, right). Thus, AuGMEnT can train networks to perform a delayed match-to-category task and it induces memory tuning for those feature variations that matter.

Probabilistic Decision Making Task

We have shown that AuGMEnT can train a single network to perform a delayed saccade/anti-saccade task or a match-to-category task and to maintain task-relevant information as persistent activity. Persistent activity in area LIP has also been related to perceptual decision making, because LIP neurons integrate sensory information over time in decision making tasks (Gold and Shadlen, 2007). Can AuGMEnT train the very same network to integrate evidence for a perceptual decision? We focused on a recent study (Yang and Shadlen, 2007) in which monkeys saw a red and a green saccade target and then four symbols that were presented successively. The four symbols provided probabilistic evidence about whether a red or green eye-movement target was baited with reward (Figure 4.4A). Some of the symbols provided strong evidence in favor of the red target (e.g. the triangle in the inset of Figure 4.4A), others strong evidence for the green target (heptagon) and other symbols provided weaker evidence. The pattern

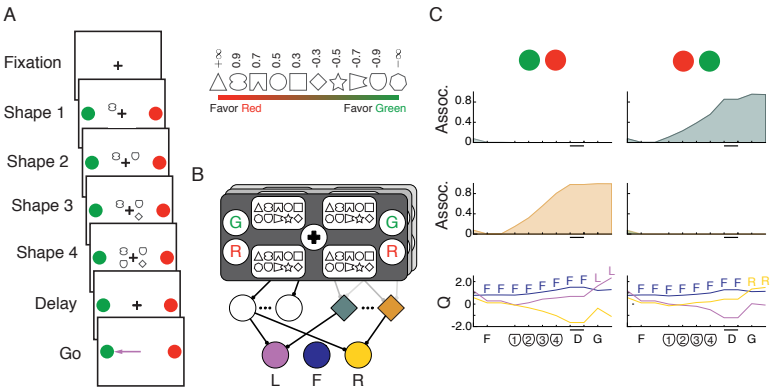


Figure 4.4: ***Probabilistic classification task.*** **A**, *After the network attained fixation, we presented four shapes in a random order at four locations. The shapes* $s_1, \ldots, s_4$ *cued a saccade to the red or green target: their location varied randomly across trials. Reward was assigned to the red target with probability* $P(R \mid s_1, s_2, s_3, s_4) = 10^W/(1+10^W)$*, with* $W = \sum_{i=1}^{4} w_i$*, and to the green target otherwise. Inset shows weights* w_i *associated with cues* s_i*.* **B**, *The sensory layer had units for the fixation point, for the colors of the targets on each side of the screen and there was a set of units for the symbols at each of the four retinotopic locations.* **C**, *Activity of two context sensitive memory units and Q-value units (bottom) in a trial where four triangle symbols were presented to a trained network. The green target is the optimal choice. F: fixation mark onset; D: memory delay; G: fixation mark offset ('Go'-signal).*

of choices revealed that the monkeys assigned high weights to symbols carrying strong evidence and lower weights to less informative ones. A previous model with only one layer of modifiable synapses could learn a simplified, linear version of this task where the symbols provided direct evidence for one of two actions (Soltani and Wang, 2009). This model used a pre-wired memory and it did not simulate the full task where symbols only carry evidence about red and green choices while the position of the red and green targets varied across trials. Here we tested if AuGMEnT could train our network with three regular and four memory units to perform the full non-linear task.

We trained the model with a shaping strategy using sequence of tasks of increasing complexity, just as in the monkey experiment (Yang and Shadlen, 2007). We will first describe the most complex version of the task. In this version, the model (Figure 4.4B) had to first direct gaze to the fixation point. After fixating for two time-steps, we gave the fixation reward r_i and presented the colored targets and also one of the 10 symbols at one of

four locations around the fixation mark, In the subsequent three time-steps we presented the additional symbols. We randomized location of the red and green targets, the position of the successively presented symbols as well as the symbol sequence over trials. There was a memory delay of two time steps after all symbols had been presented and we then removed the fixation point, as a cue to make a saccade to one of the colored targets. Reward r_f was assigned to the red target with probability $P(R \mid s_1, s_2, s_3, s_4) = 10^W/(1+10^W)$, with $W = \sum_{i=1}^{4} w_i$ (w_i is specified in Figure 4.4A, inset) and to the green target otherwise. The model's choice was considered correct if it selected the target with highest reward probability, or either target if reward probabilities were equal. However, r_f was only given if the model selected the baited target, irrespective of whether it had the highest reward probability.

The shaping strategy used for training gradually increased the set of input symbols $(2, 4, \ldots, 10)$ and sequence length $(1 \ldots 4)$ in eight steps (Table 4.3). Training started with the two 'trump' shapes which guarantee reward for the correct decision (triangle and heptagon, see Figure 4.4A, inset). We judged that the task had been learned when the success rate in the last n trials was 85%. As the number of possible input patterns grew we increased n to ensure that a significant fraction of possible input-patterns had been presented before we determined convergence (see Table 4.3). Difficulty was first increased by adding the pair of symbols with the next smaller absolute weight, until all shapes had been introduced (level 1-5) and then by increasing sequence length (level 6-8). With this shaping strategy AuGMEnT successfully trained 99 of 100 networks within a total of $500,000$ trials. Training of the model to criterion (85% correct in the final task) took a median total of $55,234$ trials across the eight difficulty levels, which is faster than the monkeys learned. After the training procedure, the memory units had learned to integrate information for either the red or green choice over the symbol sequence and maintained information about the value of this choice as persistent activity during the memory delay. Figure 4.4C shows the activity of two memory units and the Q-value units of an example network during a trial where the shield symbol was presented four times, providing strong evidence that the green target was baited with reward. The memory units became sensitive to the context determined by the position of the red and green saccade targets. The unit in the first row of Figure 4.4C integrated evidence for the green target if it appeared on the right side and the unit in the second row if the green target appeared on the left. Furthermore, the activity of these memory units ramped up gradually as

more evidence accumulated.

The activity of neurons in LIP was correlated to the log likelihood that the targets are baited (Yang and Shadlen, 2007). To investigate the influence of log likelihood on the activity of the memory units, we computed log likelihood ratio (logLR) quintiles as follows. We enumerated all 10,000 length 4 symbol combinations $s \in S$ and computed the probability of reward for a saccade to the red target, $P(R|S)$ for every combination. We next computed the conditional probabilities of reward $P(R|s_l)$ and $P(G|s_l) = 1 - P(R|s_l)$ for sequences s_l of length $l \in \{1 \dots 4\}$ (marginalizing over the unobserved symbols). We then computed $LogLR(s_l)$ as $log_{10}(P(R|s_l)/P(G|s_l))$ for each specific sequence of length l and divided those into quintiles.

To determine how the activity of memory units depended on the log likelihood that the targets were baited we first compared their average activity after observing a complete sequence of the lower and upper quintile, and reordered the quintiles so they were increasing for each unit. We then computed the average within-quintile activities over the aligned population. The upper panel of Figure 4.5A shows how the average activity of the four memory units of an example network depended on the log likelihood that the targets were baited and the lower panel shows LIP data (Yang and Shadlen, 2007) for comparison. It can be seen that the memory units' activity became correlated to the log likelihood, just like LIP neurons. Importantly, the synaptic weights from input neurons to memory cells depended on the true weights of the symbols after learning (Figure 4.5B). This correlation was also strong at the population level as can be seen in Figure 4.5C which shows the distribution of all the correlation coefficients ($N = 396$). Thus, plasticity of synapses onto the memory neurons can explain how the monkeys valuate the symbols and AuGMEnT explains how these neurons learn to integrate the most relevant information. Furthermore, our results illustrate that AuGMEnT not only trains the association units to integrate stochastic sensory evidence but also endows them with the required mixed selectivity for target color and symbol sequence that is required to solve this non-linear task (Rigotti et al., 2013).

Vibrotactile Discrimination Task

The previous simulations addressed tasks that have been employed for the study of neurons in area LIP of monkeys. Our last simulation investigated a task that has been used to study vibrotactile working memory (Hernández et al., 1997; Romo, Brody, et al., 1999). In this task, the monkey touches a key with one hand and then two vibration stimuli are applied sequentially

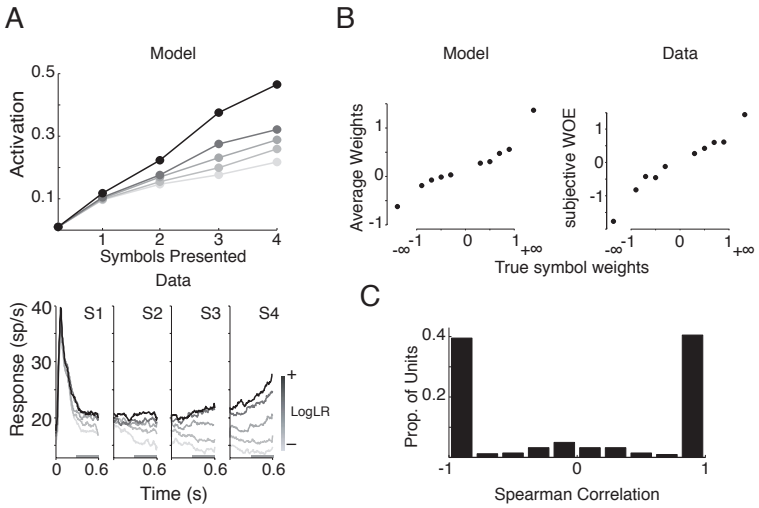


Figure 4.5: ***Tuning in the association layer in the probabilistic classification task.*** **A**, *Trials were subdivided in quintiles based on the log-likelihood ratio of the evidence favoring one target. Average activations of the four memory units of a trained model network (top;* 100,000 *trials) and LIP neurons (bottom, from Yang and Shadlen, 2007) depend on the log-likelihood ratio.* **B**, *Left, Average synaptic weights between input units representing symbols and an example memory unit are strongly correlated* ($\rho \approx 1$, $p < 10^{-6}$) *with true symbol weights. Right, Subjective weights assigned by a monkey as estimated from the performance data (from Yang and Shadlen, 2007).* **C**, *Histogram of Spearman correlations between average synaptic weights for symbols and true symbol weights for* 396 *memory units (AuGMEnT trained* 99 *of* 100 *simulated networks to criterion). Note that there are also units with zero correlation that do not contribute to the mapping of the symbols onto Q-values. These units were accompanied by other association units with stronger correlations.*

to a fingertip of the other hand (Figure 4.6A). The monkey has to indicate whether the frequency of the first vibration stimulus (F1) is higher or lower than the frequency of the second one (F2). At the end of the trial the animal indicates its choice by releasing the key and pressing one of two buttons. The overall structure of the task is similar to that of the visual tasks described above, but the feature of interest here is that it requires a comparison between two scalar values; F2 that is sensed on the finger and F1 that has to be maintained in working memory.

Recent computational work has addressed various aspects of the vibrotactile discrimination task. Several models addressed how neural

network models can store F1 and compare it to F2 (Deco, Rolls, and Romo, 2010; Machens, 2005; Miller and Wang, 2006). More recently, Barak et al. (2013) investigated the dynamics of the memory states in networks trained with three different supervised learning methods and compared them to the neuronal data. However, these previous studies did not yet address trial-and-error learning of the vibrotactile discrimination task with a biologically plausible learning rule. We therefore investigated if AuGMEnT could train the same network that had been used for LIP, with three regular units and four memory units, to solve this task. The input layer was modeled after sensory area S2 of the monkey. Neurons in this cortical area have broad tuning curves and either monotonically increase or decrease their firing rate as function of the frequency of the vibrotactile stimulus (Romo, Hernández, Zainos, and Salinas, 2003). The input units of the model had sigmoidal tuning curves:

$$r(x) = \frac{1}{1 + \exp\left(w(\theta_c - x)\right)}\,, \tag{4.33}$$

with 10 center points θ_c evenly distributed over the interval between 5.5Hz and 49.5Hz. We used a pair of units at every θ_c with one unit increasing its activity with stimulus frequency and the other one decreasing, so that there were a total of 20 input units. Parameter w determines the steepness of the tuning curve and was +/- 5. We modeled sensory noise by adding independent zero mean Gaussian noise (s.d. 7.5%) to the firing rates of the input units. We also included a binary input unit that signaled skin contact with the stimulation device (unit S in Figure 4.6B). The association and Q-value layers were identical to those of the other simulations (Figure 4.6B).

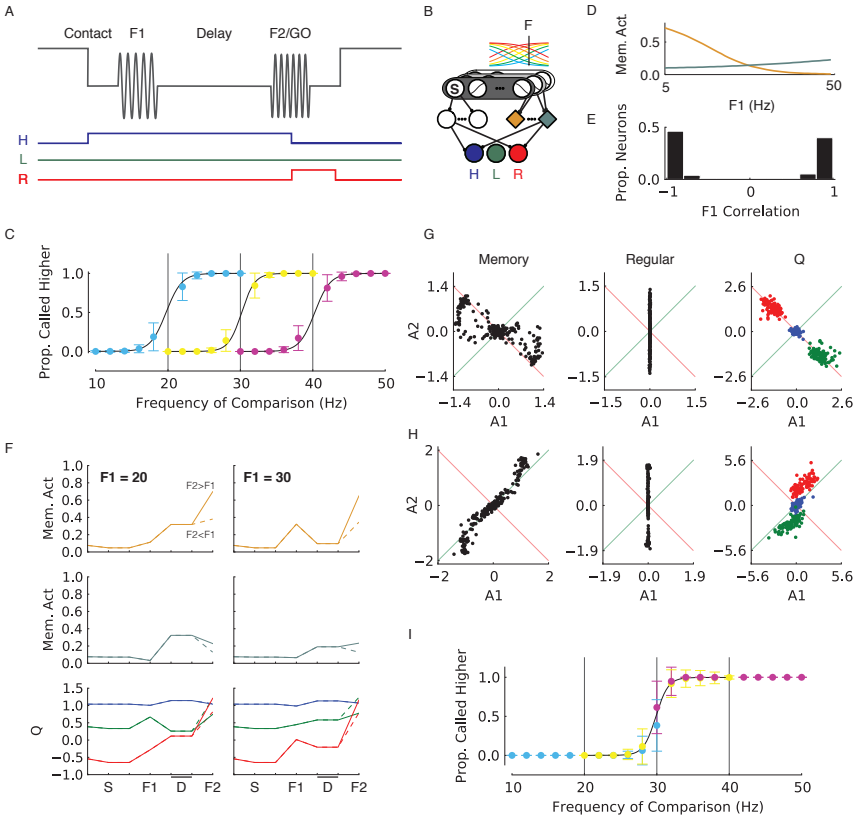

A
Contact
F1
Delay
F2/GO
H
L
R
B
F
S
H L R
D
Mem. Act
0.5
0.0
5
50
F1 (Hz)
E
Prop. Neurons
0.5
0.0
−1
0
1
F1 Correlation
C
Prop. Called Higher
1.0
0.5
0.0
10
20
30
40
50
Frequency of Comparison (Hz)
G
Memory
Regular
Q
A2
A1
1.4
0.0
−1.4
1.5
−1.5
2.6
−2.6
F
F1 = 20
F1 = 30
F2>F1
F2<F1
Mem. Act
Q
1.0
0.8
0.6
0.4
0.2
0.0
1.5
0.5
−0.5
−1.0
S
F1
D
F2
H
2
0
−2
1.9
−1.9
5.6
−5.6
I
Prop. Called Higher
Frequency of Comparison (Hz)

Figure 4.6

Figure 4.6 ***(continued from previous page): Vibrotactile discrimination task.*** **A**, *Top line shows vibrotactile stimuli, bottom colored lines show target actions for the example trial (F1 < F2). H, hold key; L, press left button to indicate F2 < F1; R, press right button to indicate F2 > F1.* **B**, *Network model. The units in the sensory layer are tuned for the tactile frequency with monotonically increasing or decreasing sigmoidal tuning curves. The binary 'S' neuron codes for skin contact of the vibrotactile probe and becomes active at 'Contact' in A.* **C**, *Average psychometric curves for* 100 *networks trained on the variable F1 task. Each set of data points (grouped by color) shows responses for the F1 stimulus that is indicated with a vertical line for flanking F2 stimuli; blue: F1=20Hz, yellow: F1=30Hz and pink: F1=40Hz. Y-axis shows the mean proportion of trials where networks indicated that F2 > F1 (each comparison was evaluated* 100 *times for every network). Error bars show s.d. over networks. Curves are logistic fits to the model responses.* **D**, *Tuning of two example memory units to F1 frequency during the delay phase.* **E**, *Histogram of linear correlations between F1 frequency and memory unit activations during the delay phase for* 100 *networks (N =* 400*).* **F**, *Example activity traces for two memory units and the three Q-value units. Left panel shows the response for F1=20Hz and F2=F1 ±5Hz (solid +5Hz, dashed −5Hz). The response of the Q-value units is coded following the color scheme in panels A and B. Right panel shows the activity of these units when F1 was* 30*Hz. F2 indicates onset of second vibration stimulus. D: Memory delay phase. Note that F2 is* 25*Hz for the continuous lines in the left panel and also for the dashed lines in the right panel, but that these trials require different responses (right button if F1=20Hz and left button if F1=30Hz).* **G**, *Scatter plot of linear regression parameters of various unit types when F2 was presented (as explained in the main text). A positive A1 (A2) parameter indicates that a unit becomes more active for higher F1 (F2). Green line shows $y = x$ and the activity of units on this line is related to the sum of F1 and F2. The red line represents $y = -x$, and the activity of units on this line represents the difference between F1 and F2. The color scheme for the Q-value units is the same as in (A) and (B).* **H**, *Scatter plot of linear regression parameters at the time of F2 presentation for networks trained on the version of the task with fixed F1.* **I**, *Psychometric curves for block-trained fixed F1 networks (see vibrotactile discrimination text). Same conventions as for (C). Only the logistic fit (black line) for F1=30Hz is drawn.*

Our first simulation addressed a version of the task where F1 varied from trial to trial (Hernández et al., 1997). A trial started when the input unit indicating skin contact with the vibrating probe became active and the model had to select the hold-key within ten time-steps, or else the trial was terminated. When the model had held the key for two time-steps, a vibration stimulus (F1, uniformly random between 5 and 50 Hz) was presented to the network for one time-step and the small shaping reward (r_i) was given. This was followed by a memory delay after which we presented the second vibration stimulus (F2), drawn from a uniform distribution between 5 and

50 Hz, but with a minimal separation of 2 Hz from F1. If F2 was lower than F1 the model had to select the left button (green Q-value unit in Figure 4.6B) - and the right button (red) otherwise - within eight time steps after the presentation of F2 to obtain the reward r_f. To determine model performance, we divided the range of F1 stimuli into 9 bins of 5 Hz and kept track of the running average of performance in 50 trials for each bin. When the model reached a performance of 80% for every F1 we disabled learning and exploration (setting learning parameters β and ϵ to zero) and checked the performance of the model for F1 stimuli of 20, 30 and 40 Hz and F2 stimuli with offsets of $[-10, -8, \ldots, -2, 2, \ldots, 8, 10]$ Hz, repeating each test 20 times. We considered learning to be successful if the model classified the nearest F2 frequencies (2 Hz distance) with a minimal accuracy of 50% and all other F2 frequencies with an accuracy better than 75%, for every F1 bin. AuGMEnT trained all 100 simulated networks to criterion within a median of $3,036$ trials. Figure 4.6C illustrates the average (±s.d.) choices of these 100 trained models as a function of F2, for three values of F1 as well as a logistic function fitted to the data (as in Hernández et al., 1997). It can be seen that the model correctly indicates whether F1 is higher or lower than F2 and that the criterion depends on the value F1, implying that the model has learned to store this analog scalar value in its working memory. What are the memory representations that emerged during learning? Figure 4.6D shows the F1 tuning of two memory units in an example network; typically the tunings are broad and can be increasing or decreasing as a function of F1, similar to what was found in experiments in the frontal cortex of monkeys (Romo and Salinas, 2003). Figure 4.6E shows the distribution of linear correlations between 400 memory units in 100 trained networks and F1 frequency; most units exhibit a strong positive or negative correlation, indicating that the networks learned to code the memory of F1 as the level of persistent firing of the memory units.

We next investigated how the model carried out the comparison process that has to take place after the presentation of F2. This comparison process depends critically on the order of presentation of the two stimuli, yet it involves information that comes in via the same sensory inputs and association units (Deco, Rolls, and Romo, 2010). We found that the memory units were indeed sensitive to both F1 and F2 in the comparison period. Figure 4.6F shows the response of two example memory units and the three Q-value units for trials with an F1 of 20 or 30 Hz, followed by an F2 with a frequency that was either 5Hz higher (solid line) or lower than F1 (dashed line). The activity of the memory units encodes F1 during the memory delay, but these units also respond to F2 so that the activity after the presentation

of F2 depends on both frequencies. The lower panel illustrates the activity of the Q-value units. The activity of the Hold Q-value unit (H, blue) is highest until the presentation of F2, causing the model to hold the key until the go-signal. This unit did not distinguish between trials that required a right or left button press. The activities of Q-value units for the left and right button press (red and green traces) explain how the network made correct decisions at the go-signal because the Q-value of the appropriate action became highest (the solid lines in Figure 4.6F show activity if F2>F1 and dashed lines F2<F1). It can be seen, for example, how the response elicited in the Q-value layer by an F2 of 25Hz depended on whether the preceding F1 was 20Hz (continuous curves in the left panel of Figure 4.6F) or 30Hz (dashed curves in the right panel). We next quantified how the activity of the memory, regular and Q-units from 100 networks (N = 400, 300 and 300 units, respectively) depended on F1 and F2 during the comparison phase with a regression (see Romo, Hernández, and Zainos, 2004) using all trials where the F2 stimulus was presented and for all combinations of the two frequencies between 5 and 50 Hz (step size 1Hz),

$$r(F_1, F_2) = F_1 a_1 + F_2 a_2 + b \tag{4.34}$$

Here a_1 and a_2 estimate the dependence of the unit's activity on F1 and F2, respectively. The activity of many memory units depended on F1 and also on F2 (Figure 4.6G, left) and the overall negative correlation between the coefficients ($r = -0.81$, $p < 10^{-6}$) indicates that units that tended to respond more strongly for increasing F1 tended to decrease their response for increasing F2 and vice versa, just as is observed in area S2, the prefrontal cortex and the medial premotor cortex of monkeys (Romo, Brody, et al., 1999; Romo, Hernández, and Zainos, 2004; Romo and Salinas, 2003). In other words, many memory units became tuned to the difference between F1 and F2 in the comparison phase, as is required by this task. In spite of the fact that F1 and F2 activate memory units with the same synapses, the inverse tuning is possible because the F1 stimulus has turned off and activated the off-cells in the sensory layer in the comparison phase. In contrast, the F2 stimulus is still 'on' in this phase of the task so that the off-units coding F2 did not yet provide their input to the memory cells. As a result, the memory units' final activity can reflect the difference between F1 and F2, as is required by the task. Regular units only have access to the current stimulus, and were therefore they are only tuned to F2 in the comparison phase (Figure 4.6G, middle). Q-value units reflect the outcome of the comparison process (Figure 4.6G, right): their correlation coefficients with F1 and F2 fall into three clusters as predicted by the required action.

The version of the task described above demanded the comparison between two flutter frequencies because F1 varied from trial to trial. Hernández et al. (1997) also studied a version of the task where F1 was fixed for a block of trials. In this version, the monkeys based their response on F2 only and did not memorize F1. As a result their performance deteriorated at the start of a new block of trials with a different F1. Networks trained with AuGMEnT also only memorize task-relevant information. Do networks trained with AuGMEnT also fail to memorize F1 if it is fixed during training? To investigate this question, we trained models with a fixed F1 of 30 Hz (Hernández et al., 1997) and presented F2 stimuli in the range between 5-50 Hz (2.5 Hz spacing) with a minimal distance from F1 of 10 Hz. We estimated convergence as the trial when accuracy reached 90% (running average of 50 trials). AuGMEnT trained all 100 networks to criterion in this simpler task within a median of 1,390 trials. After learning the fixed F1 task, we subjected the networks to block training with F1 stimuli of 20, 30 and 40 Hz (as in Hernández et al., 1997) while we presented F2 stimuli with frequencies of $[-10,-8,\ldots,-2,2,\ldots,8,10]$ Hz relative to F1 (10 total, each shown 150 times). These blocks of trials had a pseudorandom ordering but we always presented a 30Hz F1 in the last block. When we tested immediately after every block, we found that the models were well able to adapt to a specific F1. However, the models were not able to solve the variable F1 task after this extensive block training, even though they had significant exposure to different F1 stimuli. Figure 4.6I shows the average psychometric curves for 100 networks after the last block with F1=30Hz. Colors represent trials with different F1 stimuli (as in Figure 4.6C). It can be seen that the models disregarded F1 and only determined whether F2 was higher or lower than 30 Hz, just as monkeys that are trained with a blocked procedure (Hernández et al., 1997). Thus, the model can explain why the monkeys do not learn to compare the two stimuli if the F1 is fixed for longer blocks of trials. The memory units and the Q-value units now had similar rather than opposite tuning for F1 and F2 (positive correlations in the left and right panel of Figure 4.6H; compare to Figure 4.6G), which indicates that blocked training causes a failure to learn to subtract the memory trace of F1 from the representation of F2. We conclude that AuGMEnT is able to train networks on a task that requires a comparison between two analog stimuli and where the correct decision depends on stimulus order. Memory units learn to represent the analog value that needs to be memorized as a graded level of persistent activity. However, if F1 is fixed for blocks of trials, the network does not memorize F1 but learns to base its decision on F2 only, in accordance with

experimental findings.

4.5 Varying parameters of the network

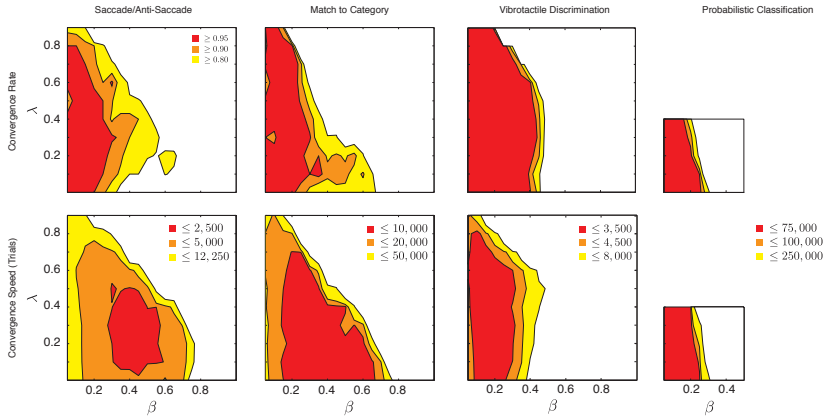


Figure 4.7: ***Robustness to variations in the parameters that control learning rate.*** *The upper row shows how the proportion of networks that converged varies as function of β (learning rate) and λ (decay of tags); white regions had a proportion of convergence lower than* 0.8*. The lower row shows the effect of β and λ on the median trial when the learning criterion was reached; white regions reached convergence lated than the yellow regions (see insets).*

It is remarkable that AuGMEnT can train the same simple network to perform a wide range of tasks, simply by delivering rewards at the appropriate times. In the simulations described above we fixed the number of units in the association layer and Q-value layer and used a single set of learning parameters. To examine the stability of the learning scheme, we also evaluated learning speed and convergence rate for various values of the learning rate β and the SARSA learning parameter λ (which determines the tag-decay parameter α because $\alpha = (1 - \lambda\gamma)$ as was explained above, γ was kept at the default value). For the saccade/antisaccade, match-to-category and vibrotactile discrimination tasks we tested $\beta \in 0.05, 0.10, \ldots, 1.0$ and $\lambda \in 0.0, 0.10, \ldots, 0.9$ while the other parameters remained the same (Tables 4.1,4.2) and ran 100 simulations for every combination. Figure 4.7 shows the proportion of networks that converged and the median convergence trial. Training in the probabilistic classification task required a number of different training stages and a longer overall training time and we evaluated this task with a

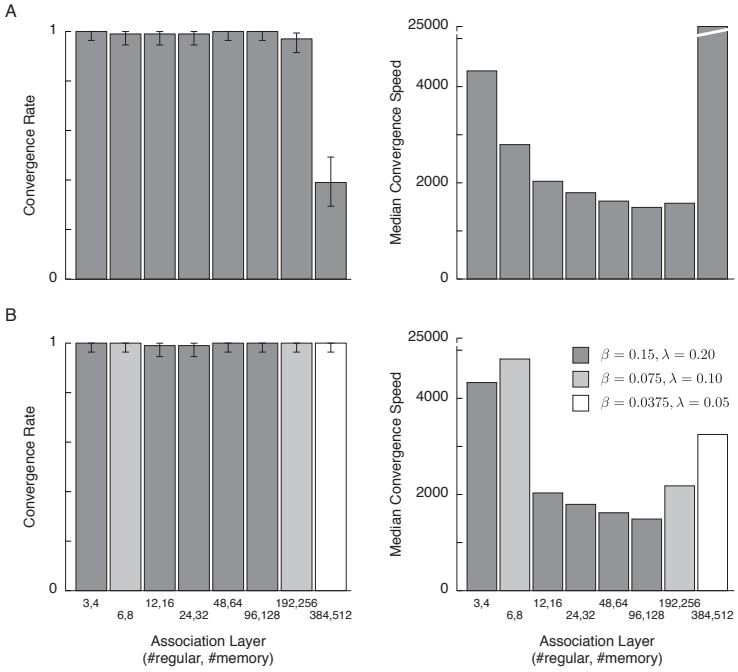


Figure 4.8: ***Varying the size of the association layer.*** **A**, *Scaling with unchanged learning parameters* β *and* λ*. Left, convergence rate (proportion of* 100 *networks that learned the saccade/antisaccade task). Error bars denote* 95% *confidence intervals. Right, median convergence speed (number of trials to criterion).* **B**, *Left, convergence rates with adjusted learning parameters. Bar shading indicates parameter setting (see legend in right panel). Right, median convergence speed with optimized parameters.*

smaller set of parameters (Figure 4.7, right). There was a wide range for the learning parameters where most of the networks converged and these ranges overlapped for the four tasks, implying that the AuGMEnT learning scheme is relatively robust and stable. So far our simulations used a fixed network with only seven units in the association layers. Can AuGMEnT also train networks with a larger association layer? To further investigate the generality of the learning scheme, we ran a series of simulations with increasing numbers of association units, multiplying the number of association units in the network described above by $2, 4, \ldots, 128$ and training 100 networks of each size in the saccade/antisaccade task. We first evaluated these larger networks without changing the learning parameters and found that the learning was largely unaffected within a limited range

of network sizes, whereas performance deteriorated for networks that were 32-128 fold larger (Figure 4.8A). The decrease in performance is likely caused by the larger number of synapses, causing larger adjustments of the Q-values after each time step than in the smaller networks. It is possible to compensate for this effect by choosing a smaller β (learning rate) and λ. We jointly scaled these parameters by 1/2,1/4 and 1/8 and selected the parameter combination which resulted in the highest convergence rate and the fastest median convergence speed for every network size (Figure 4.8B). The performance of the larger networks was at least as good as that of the network with seven units if learning parameters were scaled. Thus, AuGMEnT can also successfully train networks with a much larger association layer.

4.6 Discussion

AuGMEnT provides a new theoretical framework that can explain how neurons become tuned to relevant sensory stimuli in sequential decision tasks during trial-and- error learning. The scheme uses units inspired by transient and sustained neurons in sensory cortices (Nassi and Callaway, 2009), action-value coding neurons in frontal cortex, basal ganglia and midbrain (Hikosaka, 2005; Morris et al., 2006; Samejima et al., 2005) and neurons with mnemonic activity that integrate input in association cortex. To the best of our knowledge, AuGMEnT is the first biologically plausible learning scheme that implements SARSA in a multi-layer neural network equipped with working memory. The model is simple, yet is able to learn a wide range of difficult tasks requiring non-linear sensory-motor transformations, decision making, categorization, and working memory. AuGMEnT can train the very same network to perform either of these tasks by presenting the appropriate sensory inputs and reward contingency, and the representations it learns are similar to those found in animals trained on similar tasks. AuGMEnT is a so-called on-policy method because it only relies on the Q-values that the network experiences during learning. These on-policy methods appear to be more stable than off-policy algorithms (such as Q-learning which considers transitions not experienced by the network), if combined with neural networks (see e.g. Baird, 1995; Boyan and Moore, 1995).

AuGMEnT forms memory representations for features that need to be remembered. In the delayed saccade/anti-saccade task, training induced persistent neuronal activity tuned to the cue location and to the color of the

fixation point, but only if it was relevant. In the categorization task, units became sensitive to category boundaries and in the decision making task, units integrated sensory evidence with stronger weights for the more reliable inputs. These properties resemble those of neurons in LIP (Freedman and Assad, 2006; Gnadt and Andersen, 1988; Gottlieb and Goldberg, 1999; Yang and Shadlen, 2007) and the frontal cortex (Funahashi, Bruce, and Goldman-Rakic, 1989) of monkeys. Finally, the memory units learned to memorize and compare analog values in the vibrotactile task, just as has been observed in the frontal cortex of monkeys (Hernández et al., 1997; Romo, Brody, et al., 1999).

AuGMEnT makes a number of predictions that could be tested in future neuroscientific experiments. The first and foremost prediction is that feedback connections gate plasticity of the connections by inducing synaptic tags. Specifically, the learning scheme predicts that feedback connections are important for the induction of tags on feedforward connections from sensory cortices to the association cortex (Fig. 4.1B). A second prediction is the existence of traces in synapses onto neurons with persistent activity (i.e. memory units) that are transformed into tags upon the arrival of feedback from the response selection stage, which may occur at a later point in time. The third prediction is that these tags interact with globally released neuromodulators (e.g. dopamine or acetylcholine), which determine the strength and sign of the synaptic changes (potentiation or depression). Neurobiological evidence for the existence of these tags and their interaction with neuromodulatory substances will be discussed below. A final prediction is that stationary stimuli provide transient input to neurons with persistent activity. As a result, stimuli that are visible for a longer time do not necessarily cause a ramping of activity. In our network ramping was prevented because memory units received input from "on" and "off" input units only. We note, however, that other mechanisms such as, for example, rapidly adapting synapses onto memory cells, could achieve the same effect. In contrast, neurons in association cortex without persistent activity are predicted to receive continuous input, for as long as a stimulus is present. These specific predictions could all be tested in future neuroscientific work.

Role of attentional feedback and neuromodulators in learning

AuGMEnT implements a four-factor learning rule. The first two factors are pre- and post-synaptic activity of the units and there are two additional 'gating factors' that enable synaptic plasticity. The first gating factor is the

feedback from units in the motor layer that code the selected action. These units send an attentional signal back to earlier processing levels to tag synapses responsible for selecting this action. The importance of selective attention for learning is supported by experiments in cognitive psychology. If observers select a stimulus for an action, attention invariably shifts to this stimulus (Deubel and Schneider, 1996) and this selective attention signal gates perceptual learning so that attended objects have larger impact on future behavior (Ahissar and Hochstein, 1993; Jiang and Chun, 2001; Schoups et al., 2001). Moreover, neurophysiological studies demonstrated that such a feedback signal exists, because neurons in the motor cortex that code an action enhance the activity of upstream neurons providing input for this action (Moore and Armstrong, 2003; Roelfsema, van Ooyen, and Watanabe, 2010).

The second gating-factor that enables plasticity is a global neuromodulatory signal that broadcasts the RPE to many brain regions and determines the sign and strength of the changes in synapses that have been tagged. Dopamine is often implicated because it is released if reward expectancy increases and it influences synaptic plasticity (Montague, Hyman, and Cohen, 2004; Schultz, 2002). There is also a potential role for acetylcholine because cholinergic cells project diffusely to cortex, respond to rewards (Kilgard and Merzenich, 1998; Richardson and DeLong, 1986) and influence synaptic plasticity (Kilgard and Merzenich, 1998). Furthermore, a recent study demonstrated that serotonergic neurons also carry a reward-predicting signal and that the optogenetic activation of serotonergic neurons acts as a positive reinforcer (Liu et al., 2014). Guidance of synaptic plasticity by the combination of neuromodulatory signals and cortico-cortical feedback connections is biologically plausible because all information for the synaptic update is available at the synapse.

Synaptic tags and synaptic traces

Learning in AuGMEnT depends on synaptic tags and traces. The first step in the plasticity of a synapse onto a memory cell is the formation of a synaptic trace that persists until the end of the trial (Figure 4.1C). The second step is the conversion of the trace into a tag, when a selected motor unit feeds back to the memory cell. The final step is the release of the neuromodulator that modifies tagged synapses. The learning rule for the synapses onto the regular (i.e. non-memory) association units is similar (Figure 4.1B), but tags form directly onto active synapses, skipping the first step. We note, however, that the same learning rule is obtained if these synapses also have traces that

decay within one time-step. The hypothesis that synaptic plasticity requires a sequence of events (Friedrich, Urbanczik, and Senn, 2011; Fusi, Drew, and Abbott, 2005) is supported by the synapses' complex biochemical machinery. There is evidence for synaptic tags (Cassenaer and Laurent, 2012; Frey and Morris, 1997; Moncada et al., 2011) and recent studies have started to elucidate their identity (Moncada et al., 2011). Neuromodulatory signals influence synaptic plasticity even if released seconds or minutes later than the plasticity-inducing event (Cassenaer and Laurent, 2012; Moncada et al., 2011), which supports that they interact with some form of tag.

Comparison to previous modeling approaches

There has been substantial progress in biologically inspired reinforcement learning models with spiking neurons (Izhikevich, 2006; Potjans, Diesmann, and Morrison, 2011; Seung, 2003; Urbanczik and Senn, 2009) and with models that approximate population activity with continuous variables (Engel and Wang, 2011; Friedrich, Urbanczik, and Senn, 2011; Hoerzer, Legenstein, and Maass, 2014; Houk, Adams, and Barto, 1995; O'Reilly and Frank, 2006; Roelfsema and van Ooyen, 2005; Soltani and Wang, 2009; Suri and Schultz, 1998). Many of the models rely either on Actor-Critic learning (Sutton and Barto, 1998) or on policy gradient learning (Williams, 1992). An advantage of Actor-Critic models is that model components relate to brain regions (Houk, Adams, and Barto, 1995; Potjans, Diesmann, and Morrison, 2011; Suri and Schultz, 1998), e.g. the activity of dopamine neurons resembles the RPE. AuGMEnT has features in common with these models. For example, it uses the change in Q-value to compute the RPE (equation (4.17)). Another widely used class of models is formed by policy gradient learning methods (Seung, 2003; Williams, 1992) where units (or synapses (Seung, 2003)) act as local agents that try to increase the global reward. An advantage is that learning does not require knowledge about the influence of units on other units in the network, but a disadvantage is that the learning process does not scale well to larger networks where the correlation between local activity and the global reward is weak (Urbanczik and Senn, 2009). AuGMEnT uses 'attentional' feedback from the selected action to improve leaning (Roelfsema and van Ooyen, 2005) and it also generalizes to multi-layer networks. It thereby alleviates a limitation of many previous biologically plausible RL models, which can only train a single layer of modifiable synaptic weights and solve linear tasks (Engel and Wang, 2011; Friedrich, Urbanczik, and Senn, 2011; Houk, Adams, and Barto, 1995; Potjans, Diesmann, and Morrison, 2011; Soltani and Wang, 2009; Suri

and Schultz, 1998; Urbanczik and Senn, 2009) and binary decisions (Engel and Wang, 2011; Friedrich, Urbanczik, and Senn, 2011; Soltani and Wang, 2009; Urbanczik and Senn, 2009).

Unlike these previous models, AuGMEnT is a model of action-value learning (SARSA(λ), Sutton and Barto, 1998). It differs from many previous models in its ability to train task-relevant working memory representations, without pre-wiring. We modeled memory units as integrators, because neurons that act as integrators and maintain their activity during memory delays have been found in many cortical regions (Egorov et al., 2002; Freedman and Assad, 2006; Funahashi, Bruce, and Goldman-Rakic, 1989; Gnadt and Andersen, 1988; Gottlieb and Goldberg, 1999; Yang and Shadlen, 2007). To keep the model simple, we did not specify the mechanisms causing persistent activity, which could derive from intracellular processes, local circuit reverberations or recurrent activity in larger networks spanning cortex, thalamus and basal ganglia (Engel and Wang, 2011; Fransén et al., 2006; Koulakov et al., 2002).

A few studies included a pre-wired working memory in RL (Engel and Wang, 2011; Soltani and Wang, 2009) but there has been comparatively little work on biologically plausible learning of new memories. Earlier neural networks models used "backpropagation-through-time", but its mechanisms are biologically implausible (Zipser, 1991). The long short-term memory model (LSTM, Hochreiter and Schmidhuber, 1997) is a recent and popular approach. Working memories in LSTM rely on the persistent activity of memory units, which resemble the ones used by AuGMEnT. However, LSTM relies on the biologically implausible error-backpropagation rule. To our knowledge, only one previous model addressed the creation of working memories with a neurobiologically inspired learning scheme, the prefrontal basal-ganglia working memory model (PBWM, O'Reilly and Frank, 2006), which is part of the Leabra cognitive architecture (O'Reilly, Hazy, and Herd, 2012; O'Reilly and Munakata, 2000). Although a detailed comparison of AuGMEnT and Leabra is beyond the scope of this work, it is useful to mention a few key differences. First, the complexity and level of detail of the Leabra/PBWM framework is greater than that of AuGMEnT. The PBWM framework uses more than ten modules, each with its own dynamics and learning rules, making formal analysis difficult. We chose to keep the models trained with AuGMEnT as simple as possible, so that learning is easier to understand. AuGMEnT 's simplicity comes at a cost because many functions remained abstract (see next section). Second, the PBWM model uses a teacher that informs the model about the correct decision, i.e. it uses more information

than just reward feedback. Third, PBWM is an actor-critic architecture that learns the value of states, whereas AuGMEnT learns the value of actions. Fourth and finally, there are important differences in the mechanisms for working memory. In PBMW, memory units are bi-stable and the model is equipped with a system to gate information in prefrontal cortex via the basal ganglia. In AuGMEnT, memory units are directly activated by on- and off-units in the input layer and they have continuous activity levels. The activity profile of memory units is task-dependent in AuGMEnT. It can train memory units to integrate evidence for probabilistic decision making, to memorize analog values as graded levels of persistent activity but also to store categories with almost binary responses in a delayed match-to-category task.

Biological plausibility, biological detail and future work

We suggested that AuGMEnT is biologically plausible, but what do we mean with this statement? Our aim was to propose a learning rule based on Hebbian plasticity that is gated by two factors known to gate plasticity: a neuromodulatory signal that is released globally and codes the reward-prediction error and an attentional feedback signal that highlights the part of the network that is accountable for the outcome of an action. We showed that the combination of these two factors, which are indeed available at the level of the individual synapses, can cause changes in synaptic strength that follow gradient descent on the reward-prediction error for the transitions that the network experiences. At the same time, the present model provides only a limited degree of detail. The advantage of such a more abstract model is that it remains mathematically tractable. The downside is that more work will be needed to map the proposed mechanisms onto specific brain structures. We pointed out the correspondence between the tuning that developed in the association layer and tuning in the association cortex of monkeys. We now list a number of simplifying assumptions that we made and that will need to be alleviated by future models that incorporate more biological detail.

First, we assumed that the brain can compute the SARSA temporal difference error, which implies a comparison between the Q-value of one state-action combination to the Q-value of the next combination. Future modeling studies could include brain structures for storing the Q-value of the previously selected action while new action-values are computed. Although we do not know the set of brain structures that store action values, previous studies implicated the medial and lateral prefrontal cortex in

storing the outcome that is associated with an action (Matsumoto, 2003; Wallis, 2007). Prefrontal neurons even update the predicted outcome as new information comes in during the trial (Luk and Wallis, 2009). An alternative to storing *Q*-values is provided by Actor-Critic architectures that assign values to the various states instead of state-action combinations. They use one network to estimate state-values and another network to select actions (Houk, Adams, and Barto, 1995). Interestingly, (Houk, Adams, and Barto, 1995) proposed that the basal ganglia could compute temporal difference errors by comparing activity in the indirect pathway, which might store the predicted value of the previous time-step, and the direct pathway, which could code the predicted value of the next state. We hypothesize that a similar circuit could be used to compute SARSA temporal difference errors. In addition, we also did not model the action-selection process itself, which has been suggested to take place in the basal ganglia (see Lo and Wang, 2006).

A second simplification is that we did not constrain model units to be either inhibitory or excitatory—outgoing weights could have either sign and they could even change sign during learning. Future studies could specify more detailed network architectures with constrained weights (e.g. as in O'Reilly and Frank, 2006). Indeed, it is possible to change networks into functionally equivalent ones with excitatory and inhibitory units that have only positive weights (Parisien, Anderson, and Eliasmith, 2008), but the necessary generalization of AuGMEnT-like learning rules would require additional work.

The third major simplification is that feedback connections in AuGMEnT influence the formation of synaptic tags, but do not influence the activity of units at earlier processing levels. Future studies could include feedback connections that also influence activity of units in the lower layers and develop learning rules for the plasticity of activity propagating feedback connections. These connections might further expand the set of tasks that neural networks can master if trained by trial-and-error. In this context it is of interest that previous studies demonstrated that feedforward propagation of activity to higher cortical areas mainly utilizes the AMPA receptor, whereas feedback effects rely more on the NMDA receptor (Self et al., 2012), which plays an important role in synaptic plasticity. NMDA receptors also modify neuronal activity in lower areas, and another candidate receptor that could have a specific role in the influence of feedback connections on plasticity are metabotropic glutamate receptors, which are prominent in feedback pathways (De Pasquale and Sherman, 2011; Sherman and Guillery, 1998)

Parameter	Description	Value
β	Learning Rate	0.15
λ	Trace/Tag decay rate	0.20
γ	Discount factor	0.9
α	Tag persistence	$1 - \lambda\gamma$
ϵ	Exploration rate	0.025

Table 4.1: *Default parameters for AuGMEnT*

Architecture	Value
Input units	Task dependent
Memory units	$N = 4$
Regular units	$N = 3$
Q-value units	$N = 3$
Initial weights	Uniform $\sim [-.25, 0.25]$

Table 4.2: *Default architecture parameters for AuGMEnT*

and known to influence synaptic plasticity (Sajikumar and Korte, 2011).

A fourth simplification is that we modeled time in discrete steps and used units with scalar activity levels and differentiable activation functions. Therefore, implementations of AuGMEnT using populations of spiking neurons in continuous time deserve to be studied. We leave the integration of the necessary biological detail in AuGMEnT-like networks that would alleviate all these simplifications for future work.

Conclusions

Here we have shown that interactions between synaptic tags and neuromodulatory signals can explain how neurons in 'multiple-demand' association areas acquire mnemonic signals for apparently disparate tasks that require working memory, categorization or decision making. The finding that a single network can be trained by trial and error to perform these diverse tasks implies that these learning problems now fit into a unified reinforcement learning framework.

Task difficulty	# Input symbols	Sequence length	n trials to determine success
1	2	1	1,000
2	4	1	1,500
3	6	1	2,000
4	8	1	2,500
5	10	1	3,000
6	10	2	10,000
7	10	3	10,000
8	10	4	20,000

Table 4.3: *Probabilistic Classification convergence windows*

5 Learning attentional filtering in AuGMEnT

Abstract

Many theories propose that top-down attentional signals control activity in sensory cortices. But who controls the controller? We here investigate how AuGMEnT (Rombouts, Bohte, and Roelfsema, 2012, 2015, and chapter 4), a biologically plausible neural reinforcement learning scheme, can create higher order representations and top-down attentional signals. The learning scheme trains neural networks using two factors that gate Hebbian plasticity: (1) an attentional feedback signal from the response-selection stage to earlier processing levels; and (2) a globally available neuromodulator that encodes the reward prediction error. We demonstrate how the neural network learns to direct attention to one of two colored stimuli that are arranged in a rank-order (Lennert and Martinez-Trujillo, 2011). Like monkeys trained on this task, the network develops units that are tuned to the rank-order of the colors and it generalizes this newly learned rule to previously unseen color combinations. These results thereby provide new insights in how individuals can learn to control attention as demanded by the reward contingency. This chapter is based on Rombouts, Bohte, Martinez-Trujillo, et al. (2015).

5.1 Introduction

Our perception is highly selective. We mainly register the information that pertains to our goals and ignore the rest. Consider, for example, a tennis player waiting for the return of the ball. He focuses on the posture and motion of the opponent and on the way he holds the racquet, but he neither perceives other people on the court nor the surrounding advertisements. Through training he has learned to focus attention on the visual information that matters for the next hit. But, how did he learn to specifically attend to the visual features that matter?

In many circumstances learning depends on rewards or punishments. Winning and losing points during practice games is an incentive for

learning how to play tennis. Reinforcement learning (RL) theories provide a useful framework for understanding how feedback from the environment in the form of rewards and punishments shapes behavioral performance (Sutton and Barto, 1998). Researchers have made substantial progress in RL theories and there are influential theories about the implementation of RL in the brain (Bromberg-Martin, Matsumoto, and Hikosaka, 2010b; Dayan and Balleine, 2002). Furthermore, other influential theories have addressed how attention influences neuronal activity in visual cortical areas (Bundesen, Habekost, and Kyllingsbæk, 2005; Desimone and Duncan, 1995; Roelfsema, 2006). However, with a few exceptions (e.g. Whitehead and Ballard, 1991), these theories did yet not address the question of how the brain can learn to direct attention to those features that matter. In the present study we will investigate a new, biologically plausible RL-scheme with the aim to train a neural network to control attention. Our approach is inspired by recent findings about the influence of rewards on attention in experimental psychology and also by neurophysiological findings on how attention influences the representation of visual information in the brain.

In recent years, researchers in experimental psychology obtained many new insights in how rewards influence the deployment of attention, as documented in a recent special issue of Visual Cognition (Anderson, Theeuwes, and Chelazzi, 2015). Visual stimuli that are associated to high reward are likely to attract more attention at a later point in time than stimuli associated with lower reward (Anderson, Laurent, and Yantis, 2011; Chelazzi et al., 2013; Della Libera and Chelazzi, 2009; Hickey, Chelazzi, and Theeuwes, 2010; Raymond and O'Brien, 2009). In many tasks the reward-contingencies determine what information is relevant and what information is not, which makes it evident that reinforcers should influence attention. Neurophysiological studies have demonstrated that areas of the parietal and frontal cortex selectively represent task-relevant information, and the pairing of stimuli with rewards and punishments should therefore favor the representation of these stimuli (Duncan, 2010; Gottlieb and Balan, 2010). Most theories of attention state that these neurons in frontal and parietal cortex provide a top-down signal that influences the representation of stimuli in the visual cortex (Corbetta and Shulman, 2002; Desimone and Duncan, 1995; Miller and Cohen, 2001). Indeed, the neuronal responses elicited by attended objects are enhanced in many, if not all, areas of visual cortex (Reynolds and Chelazzi, 2004; Treue and Maunsell, 1996), and attentional selection processes can be monitored even at the level of the primary visual cortex (area V1) (Roelfsema, 2006). Because the required top-down attentional control

signals depend on the precise task demands, they should be strongly influenced by RL. However, the mechanisms that allow reward signals to shape these attentional top-down control signals are not well understood.

Interestingly, neuronal activity evoked by stimuli associated with high rewards is also stronger in visual and association cortex than activity evoked by stimuli associated with less reward (Louie, Grattan, and Glimcher, 2011; Pastor-Bernier and Cisek, 2011; Serences, 2008; Stănişor et al., 2013). Although some have proposed that the effects of attentional selection and reward expectancy differ at the neuronal level (Louie, Grattan, and Glimcher, 2011; Platt and Glimcher, 1999), a recent study demonstrated that V1 neurons modulated by reward expectancy are also modulated by attention and with a similar timing (Stănişor et al., 2013). The latter suggests both modulations are caused by a common top-down signal reaching the visual cortex driven by both reward expectation and selective attention (Maunsell, 2004).

To gain more insight in how reinforcers can influence the deployment of attention, we will here train a neural network model to carry out a non-linear attentional control task that has been studied in monkeys by Lennert and Martinez-Trujillo (2011). We will train the network with AuGMEnT (Rombouts, Bohte, and Roelfsema, 2012, 2015, and chapter 4), which incorporates two factors that are known to modulate synaptic plasticity (Roelfsema and van Ooyen, 2005; Roelfsema, van Ooyen, and Watanabe, 2010). The first factor is a reward-prediction error that codes whether the outcome of an action is better or worse than expected. It is thought that such a reward prediction error is broadcast throughout the brain by the release of neuromodulators, such as dopamine or serotonin, so that it is available at many synapses and can influence their plasticity (Liu et al., 2014; Schultz, 2002). The second factor is an attentional top-down signal from the response selection stage to earlier processing levels. The learning rule enforces that this top-down signal highlights the subset of synapses that are responsible for the action that was chosen by the network and this signal thus determines which synapses are sensitive to the neuromodulators that determine the changes in synaptic strength. Overall, the learning rule incorporates four signals that are all available locally, at the synapse: (i) presynaptic activity; (ii) postsynaptic activity; (iii) the globally released neuromodulatory signal and (iv) activity of feedback connections from the response selection stage.

In the task of Lennert and Martinez-Trujillo (2011) the monkeys first had to direct their gaze to a fixation mark in the center of a display flanked by two moving random dot patterns (RDPs) made up of grey dots

(Figure 5.1A). After a delay, both RDPs changed color. These color changes were an attentional cue for the monkeys, because they indicated which of the patterns was the target and which one the distracter (e.g., green was the target and red the distracter). The monkeys' task was to respond to a brief change in the motion direction of the target pattern by releasing a button while ignoring a similar change in the distracter's direction. The crucial design feature of the task is that that were six possible colors, which were organized in a rank order following an ordinal scale (red < orange < yellow < green < blue < purple; the original study used different colors which did not map onto the spectrum in this orderly manner). The monkeys had to attend to the pattern with the highest color rank (the target) and ignore the pattern with the lowest rank (the distracter). They only received a juice reward for responses to a change in the target direction, whereas trials were aborted without reward if they responded to the distracter change. Note that this task is non-linear, because the color cues determine whether a response is required to the left motion cue so that the right motion cue should be ignored, or whether the contingency is reversed. The monkeys were found to learn the order relationships by trial and error, which is remarkable because they saw only two colored patterns at the same time. They generalized the rule to infer the relative rank of new color pairs that they had not seen during training (i.e., transitive knowledge).

Once the monkeys had learned this task Lennert and Martinez-Trujillo (2011) recorded the activity of neurons in the dorsolateral prefrontal cortex. They observed that the firing rate of a substantial fraction of the cells coded the location of the target pattern. Some of these neurons increased their firing rate if the target pattern was on the left side of the display, whereas others increased their response if the target pattern was on the right. Moreover, the strength of the attentional control signal depended on the distance between the ranks of the target and distracter colors. The signal was strongest if the distance between the ranks was high and weaker for colors with adjacent ranks (distance effect).

Although the study of RL was not the aim of Lennert and Martinez-Trujillo (2011), as rewards were only used as incentive for the monkeys to do the task, this study does allow us to address the central theoretical issues that we wish to investigate: how do brain structures control attention and how do they optimize this control while animals learn a task by RL? More specifically: (1) which mechanism can cause neurons in prefrontal cortex to encode the rank order of the colors? (2) how do these control signals ensure that the monkey only responds to changes in the direction of the target? and (3) how do the monkeys generalize the rule to new colors that they

have not seen during training?

To address these questions, we used the AuGMEnT learning scheme that can train neural networks to perform many of the tasks that have been used in monkey studies, including tasks that require decision-making, non-linear sensory-motor mappings, working memory and categorization (Rombouts, Bohte, and Roelfsema, 2012, 2015). The network aims to learn the value of the successive actions that need to be taken during a trial, such as holding or releasing a button. These action values correspond to the expectancy of obtaining a reward at the end of the trial, and the representation of these action values allows the model to make the optimal choice at every time step during a trial.

A unique feature of the learning rule is that neuronal plasticity makes feedforward and feedback connections reciprocal, in accordance with anatomical and neurophysiological findings (Felleman and Van Essen, 1991; Mao et al., 2011). When the model learns to select actions based on the relevant features, the units coding these features start to receive attentional feedback from the response selection stage. We show that a model trained with AuGMEnT can indeed learn the attentional control task, and that the behavior of the model is similar to that of monkeys. We further show that the model explains the formation of rank-difference tuning and how trial-and-error learning can shape attentional top-down signals.

5.2 Methods

Model

We have described the AuGMEnT learning rule in previous work (Rombouts, Bohte, and Roelfsema, 2012, 2015, and chapter 4), but we have not used it so far to study how trial-and-error learning shapes top-down attentional selection signals. In these previous studies we outlined the theory behind the model and how it optimizes network performance. We also demonstrated how AuGMEnT allows networks to learn non-linear sensory-motor transformations and decision-making tasks. We will here summarize the key features of the model, which are necessary to understand learning of the attentional control task. If you are already familiar with AuGMEnT, you can safely skip to the next paragraph.

An important property of AuGMEnT is that it can train a two-layer neural network to perform many tasks, by simply varying the input stimuli and the reward contingency. In the present study, we used the same network

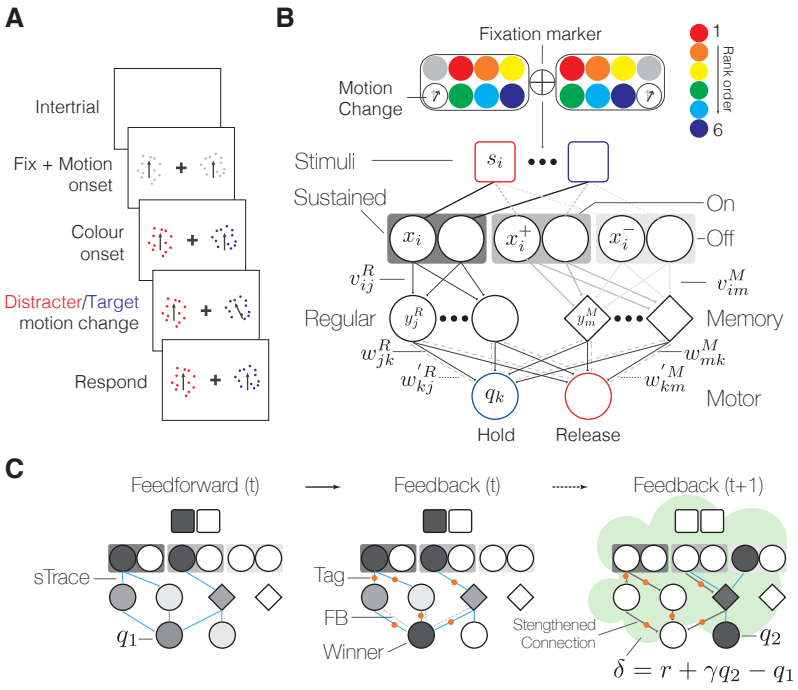


Figure 5.1: ***Task and Model.*** **A**, *When the monkeys attained fixation, two grey moving stimuli appeared. After a delay the stimuli were colored and these colors determined which of the two stimuli had to be monitored for a motion change. After a motion change in the relevant stimulus, the monkey had to release a button whereas motion changes at the irrelevant side had to be ignored.* **B**, *The neural network was composed of an input layer with sustained, on and off units, an association layer with regular and memory units and a motor layer with units coding for action values (Q-values). Motor units have feedback connections (dashed) to the association layer.* **C**, *Sensory stimuli give rise to activations in the top layer of the network. Synaptic traces (blue lines) are formed on synapses that have feedforward activations (left). Then a stochastic WTA process in the motor layer selects one of the actions for execution (middle). The selected action unit informs the rest of the network that is was selected via feedback connections (dashed lines in the middle panel). The interaction of feedback signals from the selected action and feedforward signals arising from the sensory stimuli give rise to the formation of synaptic tags (orange hexagons). These tags label the synapses in the network that were responsible for the selected action (middle). Finally, after the actually executing the action selected at time t, observing a reward and selecting a new action for execution, the difference between the predicted value at time t and the value of the actually observed transition gives rise to a prediction error δ, which is encoded by a globally released neuromodulator (green cloud). Synaptic weights in the network are updated by a simple multiplicative interaction of the tag strength and the prediction error signal (right).*

topology as in our previous work (Figure 5.1B) to understand how rewards influence attentional control. The model is a two layer neural network that learns to predict action values (also known as Q-values, Sutton and Barto, 1998) for the different actions that it can take (Figure 5.1B). Thus, when a stimulus is presented to the input layer, the model's task is to propagate activity from the input layer to the association layer and then to the Q-value layer to compute the value of the different actions that the model can take. Phrased more formally, there is a Q-value unit for every possible action a and this unit aims to represent the (discounted) expected reward for the remainder of a trial if the network selects an action a in the current state s:

$$Q^{\pi}(s,a) = E_{\pi}[R_t | s_t = s, a_t = a], \qquad \text{with} \qquad R_t = \sum_{p=0}^{\infty} \gamma^p r_{(t+p+1)}, \tag{5.1}$$

where $E_{\pi}[\cdot]$ is the expected value of the sum of discounted future rewards R_t, given current action-selection policy π, and where $\gamma \in [0,1]$ determines the discounting of future rewards r. Discounting means that rewards in the distant future are considered less valuable than rewards that can be earned at earlier time points.

Learning is guided by a neuromodulatory signal that represents the SARSA temporal difference error δ:

$$\delta(t) = r(t) + \gamma Q'(t) - Q(t-1), \tag{5.2}$$

where $r(t)$ is the scalar reward observed on time step t. SARSA is a method from the RL literature (Rummery and Niranjan, 1994; Sutton and Barto, 1998) that considers transitions from one state-action combination to the next while evaluating the reward (hence SARSA; State-Action Reward State-Action). This δ is positive if the outcome of an action was better than expected and negative if it was worse. For example, if the Q-value of an action taken on time-step t equals 0.8, the network expects 0.8 units reward for the remainder of the trial. If the network at time $t+1$ selects the next action with a value of 0.9, the action at time t turned out to be better than expected ($\delta = 0.1$) and the network updates the synapses to increase the value of the action that was taken at time t[1].

The sensory layer ('stimuli' in Figure 5.1B) of the network represents the current stimulus with three different unit types: Instantaneous ($x_i(t)$) units, and On ($+, x_i^+(t)$) and Off ($-, x_i^-(t)$) units, so that each sensory input

[1] Note that this holds for the case where $\gamma = 1$.

$s_i(t)$ is encoded by three different types of input units:

$$x_i(t) = s_i(t)\,, \tag{5.3}$$

$$x_i^+(t) = [s_i(t) - s_i(t-1)]_+\,, \tag{5.4}$$

$$x_i^-(t) = [s_i(t-1) - s_i(t)]_+\,, \tag{5.5}$$

where $[\cdot]_+$ is a threshold operator that leaves positive inputs unchanged but returns 0 for negative inputs. Thus, instantaneous units code for the current sensory input, whereas On-units are active for only one time-step if a feature has just appeared and Off-units when it disappeared.

The association (middle) layer of the network (middle in Figure 5.1B) is equipped with two different types of units: regular units and memory units. The activity of regular units depends on the current activity of units in the input layer, whereas memory units exhibit persistent activity so that they can represent working memories of stimuli presented earlier, as are found in for instance in prefrontal and parietal cortex (Funahashi, Bruce, and Goldman-Rakic, 1989; Gnadt and Andersen, 1988). We will first describe the activity of regular units before we describe the activity of the memory units.

Instantaneous units x_i in the input layer project to the regular association units via synaptic weights v_{ij}^R (with v_{0j}^R is a bias weight) (Figure 5.1B). Their activity y_j^R is determined by:

$$inp_j^R(t) = \sum_i v_{ij}^R x_i(t)\,, \tag{5.6}$$

$$y_j^R(t) = \sigma(inp_j^R(t)) \equiv 1/(1 + \exp(\theta - inp_j^R(t))\,, \tag{5.7}$$

where $\sigma(\cdot)$ is a non-linear activation function (squashing function) and we note that our results generalize to other forms of this activation function. The activation function of the memory units is similar. The on (+) and off (−) units in the sensory layer project to the memory units via synaptic weights v_{im}^+ and v_{im}^- and their activation is determined as:

$$inp_m^M(t) = inp_m^M(t-1) + \sum_i (v_{im}^+ x_i^+(t) + v_{im}^- x_i^-(t))\,, \tag{5.8}$$

$$y_m^M(t) = \sigma(inp_m^M(t))\,, \tag{5.9}$$

so that their activity also depends on features that have appeared or disappeared during earlier time steps in the trial. The instantaneous units in the input layer do not project to the memory units to prevent the integration of

a constant input, which would give rise to ramping activity of the memory units.

Finally, the regular and memory association units project to the output/motor layer via synaptic weights w^R_{jk} (with w^R_{0k} as a bias weight) and w^M_{mk}, respectively, to give rise to a set of Q-values q_k for the different actions k that the model can take so that:

$$q_k(t) = \sum_m w^M_{mk} y^M_m(t) + \sum_j w^R_{jk} y^R_j(t)\,. \tag{5.10}$$

Thus, when activity has been propagated to the output layer, this layer encodes a set of Q-values, one for every action. Then a stochastic winner take all (WTA) competition determines the action that the network will perform. With high probability $(1-\epsilon)$ the greedy action (with highest Q) is selected (ties broken randomly), but with a small probability ϵ the winning action is determined by sampling from the Boltzmann distribution to allow the exploration of other actions:

$$P_B(k) = \frac{\exp(q_k)}{\sum_{k'} \exp(q_{k'})}\,, \tag{5.11}$$

where $P_B(k)$ is the probability that action k is selected. After selecting an action a, the activation in the Q-value layer becomes $z_k = I_{ka}$, where I_{ka} is an identity function returning 1 if $k = a$ and 0 otherwise. In other words, only the winning output unit a has activity 1 and the activity of the other units in the output layer becomes zero. The output layer then informs the rest of the network about the selected action via feedback weights w'^M_{km} to memory units and w'^R_{kj} to regular association units (dashed lines in Fig. 1B), and the interaction of the feedback and feedforward activations is used to constrain synaptic plasticity to those synapses that were actually involved in selecting the action. The network creates synaptic traces and synaptic tags that determine synaptic plasticity. The traces signal that a synapse has been active, whereas tags signal that the synapse was involved in the selection of an action, so that the synapse is going to be held responsible for the outcome of this action. Synaptic traces on synapses from regular and memory units to the output layer are proportional to the strength of the afferent signal through these synapses:

$$sTrace^R_{jk}(t) = y_j(t)\,, \tag{5.12}$$

$$sTrace^M_{mk}(t) = y_m(t)\,. \tag{5.13}$$

These traces are a prerequisite for plasticity because tags can only form on those synapses that contain a trace. Tags in the output layer form only on

synapses onto the winning output unit that was selected for an action, and they then decay exponentially:

$$Tag^R_{jk}(t+1) = \lambda\gamma Tag^R_{jk}(t) + sTrace^R_{jk}(t)z_k(t)\,, \tag{5.14}$$

$$Tag^M_{mk}(t+1) = \lambda\gamma Tag^M_{mk}(t) + sTrace^M_{mk}(t)z_k(t)\,, \tag{5.15}$$

The parameter $\lambda \in [0,1]$ determines the rate of decay of tags (in RL these tags are called eligibility traces; Sutton and Barto, 1998). The strength of the tag determines the degree of plasticity of the synapse.

The updates for the synapses v^R_{ij} between the input and association layer have a similar form:

$$sTrace^R_{ij}(t) = x_i(t)\,, \tag{5.16}$$

$$Tag^R_{ij}(t+1) = \lambda\gamma Tag^R_{ij}(t) + sTrace^R_{ij}(t) fb^R_j(t)\,, \tag{5.17}$$

$$= \lambda\gamma Tag^R_{ij}(t) + sTrace^R_{ij}(t)\sigma'(inp^R_j(t))w^{'R}_{aj}\,, \tag{5.18}$$

Note that the formation of tags here depends on $fb^R_j(t)$, which is a shorthand for the attentional feedback signal that originates from the winning output unit a and arrives at unit j through the feedback connections $w^{'R}_{aj}$, and σ' is the derivative of the activation function with respect to its input, which has the convenient form $\sigma'(inp^R_j) = \sigma(inp^R_j)(1-\sigma(inp^R_j))$. Thus, only synapses v^R_{ij} that receive feedback from the response selection stage are rendered plastic because they form tags. The updates for the synapses $v^{+/-}_{im}$ onto memory units are similar:

$$sTrace^{+/-}_{im}(t) = sTrace^{+/-}_{im}(t-1) + x^{+/-}_i(t)\,, \tag{5.19}$$

$$Tag^{+/-}_{im}(t+1) = \lambda\gamma Tag^{+/-}_{im}(t) + sTrace^{+/-}_{im}(t) fb^M_m(t)\,, \tag{5.20}$$

$$= \lambda\gamma Tag^{+/-}_{im}(t) + sTrace^{+/-}_{im}(t)\sigma'(inp^M_m(t))w^{'M}_{am}\,, \tag{5.21}$$

where the critical difference is that synaptic traces to memory units accumulate, i.e. they reflect the total input provided by a synapse during the trial, whereas all other traces disappear after a single time step. After executing action a with expected value q_a and updating the traces and tags as specified above, the network makes a transition to a new state in the environment and it selects a new action a' with associated Q-value q'_a on the next time step. The network may also receive a reward r during this transition. After the transition, a neuromodulatory substance (e.g. dopamine) is globally released in the network, which encodes the SARSA prediction error $\delta(t)$. The concentration of the neuromodulator depends on the differ-

ence between successive Q-values, taking the discounting as well as the reward r into account:

$$\delta(t) = r(t) + \gamma q_{a'}(t) - q_a(t-1)\,. \tag{5.22}$$

This prediction error is positive if the outcome of the previous action is better than expected and negative if it is worse, and it is also positive if the network experiences a transition to a higher Q-value but does not receive an immediate reward. Synaptic plasticity in the network is then simply determined by the interaction of this neuromodulatory substance with the tagged synapses as:

$$\Delta w^R_{jk} = \beta\delta(t) Tag^R_{jk}(t)\,; \qquad \Delta w^M_{lk} = \beta\delta(t) Tag^M_{lk}(t)\,, \tag{5.23}$$

$$\Delta v^R_{ij} = \beta\delta(t) Tag^R_{ij}(t)\,; \qquad \Delta v^{+/-}_{im} = \beta\delta(t) Tag^{+/-}_{im}(t)\,, \tag{5.24}$$

where β determines the learning rate. Feedback weights are updated in the same manner. As was mentioned in the introduction, the learning rule is neurobiologically plausible because the factors that determine plasticity are the pre- and postsynaptic activity, the "attentional" feedback signal from the response selection stage and the neuromodulator coding for $\delta(t)$, signals that are all available locally, at the synapse.

When the network reaches the end of a trial the γ parameter used in the computation of the reward prediction error is set to zero for the corresponding time-step and the dynamic parameters (i.e. tags, synaptic traces, unit activations) in the network are cleared. It can be shown that the above learning rules change the synapses in the network as to minimize the SARSA temporal difference prediction errors by stochastic gradient descent, and that AuGMEnT can be seen as a biologically plausible implementation of the SARSA(λ) learning algorithm, extended with a working memory (chapter 4 and Rombouts, Bohte, and Roelfsema, 2012, 2015). The non-linear units of the association layer allow the network to learn non-linear mappings from sensory stimuli onto Q-values, as is essential in the present attentional control task. Figure 5.1C provides a graphical summary of the learning mechanism described above.

Color task network architecture In order to investigate how AuGMEnT can train a network to control attention, we constructed the simplest possible network that captures the essentials of the attentional control task based on color ranking. We equipped the network with a retinotopically organized sensory layer (Figure 5.1B, top) with binary neurons, i.e. neurons that were either active or silent, representing the seven possible colors on

the left side and on the right side. We additionally included a binary neuron that represented the change in motion direction on each side, which the model had to report (for targets) or to ignore (for distracters) and a binary unit for the fixation mark in the middle. As in our previous work (Rombouts, Bohte, and Roelfsema, 2012, 2015, and chapter 4), the association layer was equipped with three regular units and four memory units. We previously demonstrated that AuGMEnT can also be used to train networks with many more units in the association layer. The action layer of the network had neurons that coded for two different actions: one to hold a response button and the other to release the button. For all results reported here we used the following parameters: $\beta = 0.35, \lambda = 0.40, \gamma = 0.9$ and $\epsilon = 0.025$ and $\theta = 2.5$. We note that the results below do not critically depend on the precise value of these parameters—we found AuGMEnT to be robust across a wide range of parameters and in a large variety of tasks (Rombouts, Bohte, and Roelfsema, 2015, and chapter 4).

Model of the attentional control task

The task (Figure 5.1A) was modeled as a sequence of discrete time steps. Every trial started with an empty screen, shown for one time step. Then we presented the fixation mark flanked by the grey patterns. The model had to "press" the response button within ten time steps, otherwise the trial was terminated without reward. If the model "held" the button for two time-steps, the colors of the stimuli changed, with the target changing to a color with a higher rank than the distracter. If the target stimulus changed direction (i.e. when the binary 'change' neuron on the corresponding side became active) the model had to "choose" the release action within eight time steps to receive a "reward" of 1.5 units. However, if the distracter stimulus changed direction, the model had to hold the button for two additional time steps, after which the target stimulus would briefly change, indicating that the model could release the button to obtain the reward. As in our earlier work (Rombouts, Bohte, and Roelfsema, 2015, and chapter 4), we used a shaping strategy to encourage the model to learn to hold the button by giving a small reward (hold-reward) of 0.2 units when the model held the button for two time-steps after the fixation mark turned on. We note that this shaping strategy is not essential for the learning of the task, but that it speeds up the learning process, as in animal learning (Krueger and Dayan, 2009).

Model training

We carried out two separate sets of simulations. In the first set of simulations we investigated the generalization performance of the network by training it on a subset of all color combinations and then testing performance for color combinations that had not been presented during training. This allowed us to investigate whether AuGMEnT can generalize to unseen combinations of colors, as monkeys did. In the second set of simulations we presented all color combinations from the start of training. The monkeys in Lennert and Martinez-Trujillo (2011) were trained for 3-5 months on this full version of the task, and these simulations allowed us to investigate the neuronal tuning that develops after significant experience in the color task.

Test of generalization

The first training scheme was developed to study if the model could generalize the color-ranking scheme to unseen combinations of colors. We used the same shaping scheme as used to test the generalization performance of the monkeys in the Lennert and Martinez-Trujillo (2011) study. The models were first trained on the color-pairs 'green-blue', 'yellow-green' and 'yellow-blue' (rule: yellow < green < blue). The ordering of the colors and the stimulus (target or distracter) where the first direction change occurred were randomized. We will use the term 'respond trial' for a trial where the target stimulus changed first, and 'ignore trial' for a trial where the distracter stimulus was the first to change. After the model learned the task (see below) we sequentially trained the model on unseen colors, first training on the combination 'red-yellow'. In combination with the initial set, the relative ranking of the red color could be inferred from the 'red-yellow' pair, because red < yellow, and yellow < green < blue. If the model inferred this ordering, it should learn the other combinations with the red color ('red-green' and 'red-blue') more easily. We repeated this training scheme for the 'orange' color, first training on 'orange-yellow', and then testing learning speed for 'orange-green', 'orange-blue' and 'red-orange' simultaneously. We considered that learning was complete when the model made 85% correct choices in the last 100 trials of each color pair. All networks learned the task within a median of 1,800 trials.

Training on all color pairs

We also evaluated the model's performance when it was immediately exposed to all color pairs, without shaping except for the small hold-reward.

The locations of colors and the distracter/target trials were generated uniformly at random. We considered learning successful if the models made more than 85% correct choices in the last 100 presentations of all possible color pairs, collapsing over the two possible locations of the two colors and across respond and ignore trials. All models learned this task in less than 5,200 trials.

5.3 Results

We investigated the learning behavior of AuGMEnT using two different training schemes as explained in the methods section above. In the original study, the monkeys were either trained on a version of the task that tested their generalization to new color combinations or in the full task with all color combinations. In accordance with these two training schemes, we used a first scheme to test if a neural network trained with AuGMEnT would generalize the rule to new color-pairs, and a second scheme to study the behavior of networks that have been trained with all color combinations.

Generalization to new color combinations

In the first version of the task we trained 100 networks (i.e. repetitions with different initializations of the network weights) to test generalization (see Methods), aiming to record the decisions that the models made during each stage of the training procedure. In the first phase, we trained the networks to discover the general rule with three basic color pairs, 'green-blue', 'yellow-green' and 'yellow-blue'. The only feedback that the network received about its performance was the small hold-reward if it held the response button for two time-steps and the large reward if it responded to the motion change on the relevant side. In spite of this limited feedback about their performance, all networks learned these patterns within a median of 1,800 trials, which is fast when compared with learning by the monkeys.

Figure 5.2 shows the distributions of the number of mistakes that the models made in each learning stage for the different color pairs, including trials where the model released the button prematurely, before the first change in the direction of the moving patterns. After the models had learned the ranking of green, blue and yellow they rapidly generalized to previously unseen color pairs. Specifically, 'red-yellow' was clearly the most difficult to learn, because the color red was introduced for the first time and its rank relative to yellow was unknown. Once the models had learned that red had a lower rank than yellow, the subsequent transfer to 'red-green' and

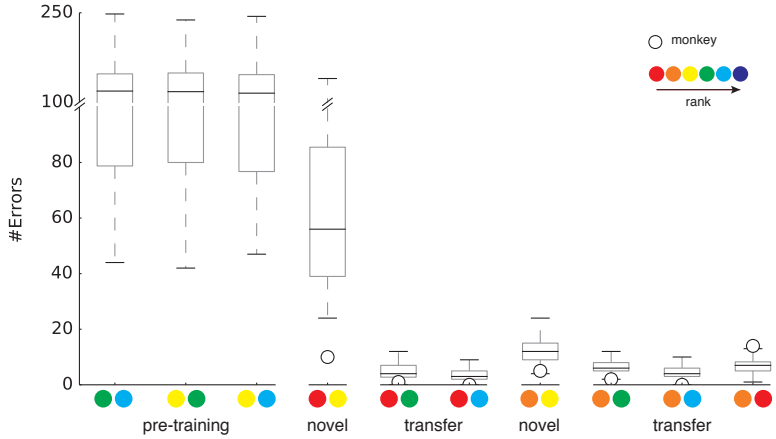


Figure 5.2: ***Generalization of the model to new color combinations.*** *The box plots illustrate the number of errors made, on average, by* 100 *networks trained on the generalization version of the color task. First, the models were trained on three color pairs; green-blue, yellow-green and yellow-blue. Then, models were exposed to red as a novel color, which initially only occurred in combination with the known color yellow. After reaching criterion performance (methods), transfer learning was tested by training the model on the remaining color pairs, red-green and red-blue. It can be seen that few additional errors were made for these new color pairs. This scheme was repeated for the orange color. The error pattern for a monkey trained using the same scheme is overlaid (white circles). The boxes illustrate the lower quartile, median (thick) and upper quartile, whereas the whiskers extend to most extreme data point within* 1.5 *multiples of the inner quartile range). Note the discontinuity of the y-axis.*

'red-blue' was easy, consistent with the hypothesis that the model could exploit the order relationship, as yellow was lower in rank than green and blue, so that red<yellow implies red<green and red<blue. We subsequently introduced the orange color, pairing it with yellow. The model quickly learned that orange had a lower rank than yellow and it subsequently made only few errors with 'orange-green' and 'orange-blue'. The error rate increased slightly for the last color pair ('orange-red'), but this also occurred when a monkey was trained on this task. This phenomenon can be explained because the relative rank of orange and red is undetermined when the model has learned that orange<yellow and red<yellow. The general learning pattern of the networks follows the monkeys' behavior (white circles in Figure 5.2). Indeed, the Spearman rank correlation between the median network performance and the monkey's performance was 0.86 ($p < 0.012$). Thus, these simulations indicate that neural networks trained

with AuGMEnT generalize a color ranking scheme to unseen combinations of colors, and that the pattern of errors is similar to that shown by a monkey when trained on the same task.

Full task

We next investigated the behavior of AuGMEnT on the full task with all possible combinations of color stimuli, which was also used to train the monkeys. For these simulations we trained an additional 100 networks. They all managed to learn the task to criterion within a median of 2,800 trials, which is fast compared to the monkeys who took about 3-5 months of training. Note that the learning of the full task was slower than in the generalization task (1,800 trials). This slightly slower learning process can be explained by the fact that the purple (highest rank) color was not included in the generalization task. Furthermore, the generalization task included many examples with adjacent ranks, which is helpful if the task is to infer a rank order (Krueger and Dayan, 2009).

Model accuracy as function of color distance

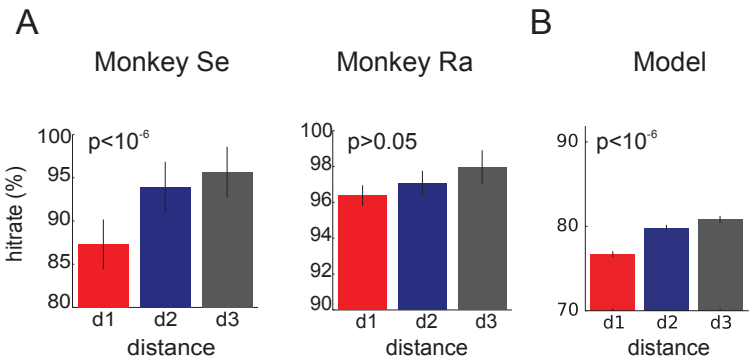


Figure 5.3: ***Effect of difference in color rank on the error rate.*** **A**, *Hit rate of monkeys Ra and Se after 3-5 months of training as a function of distance between color ranks. Error bars denote s.e.m.* **B**, *Average hit rates of models ($N = 100$) throughout learning the full color rank task as a function of distance between color ranks. Error bars show s.e.m. The networks were trained until they reached an accuracy of* 85% *(see methods), which explains why the total performance is around* 80% *when averaged over the whole training period. Note that the scale of panels A and B differs.*

When a task requires the comparison of stimuli that are "close" in rank, humans and animals tend to require a longer time to reach a decision and make more errors (Dehaene, Dehaene-Lambertz, and Cohen, 1998). Also the monkeys that were trained on the color-rank task exhibited a clear effect of the difference in rank between the color stimuli on the error rate (Figure 5.3A, Lennert and Martinez-Trujillo, 2011).

To investigate whether the networks trained with AuGMEnT exhibit a similar sensitivity, we recorded all errors made by the networks during training, for colors separated by distance of 1, 2, or 3 on the color scale. Figure 5.3B shows that networks trained with AuGMEnT exhibited a similar distance effect (one-way analysis of variance-ANOVA, Kruskal-Wallis post-hoc test, $H = 45.76, p < 0.001$). Thus, the neural networks captured many aspects of the behavioral performance of the monkeys. To examine how the networks learned to focus their attention on the side of the color with highest rank, we next examined the activity of the units in networks trained to perform the full task.

Activity of the units in a trained network

To obtain a first intuition of how the trained networks solve this task, Figure 5.4 shows the activity of two memory units in the association layer and also the activity of the Q-value units in the output layer of an example network, for all trial types with a green and orange stimulus. The unit on the top (light blue trace) had a stronger response when the target color was on the left (compare the first and third column) whereas the unit in the middle row (grey trace) had a stronger response when the target color was on the right. It can be seen that motion stimuli also had strong effects on the units' activity level; for instance the blue "left" unit received excitatory input from a motion change on the left (M1 in the first column of Figure 5.4 and M2 in the second column). Similarly, the grey "right" unit was excited by the right motion stimulus. Examination of the activity of Q-value units (lower row in Figure 5.4) revealed that the Q-value of the "Hold" action is higher than the activity of the "Release" action, until the moment where the motion stimulus is presented on the side that needs to be monitored by the model. This activity pattern of the Q-value units follows from the fact that this example network had learned to hold the lever until the motion change occurred on the relevant side. When we examined the activity of many networks, we found that memory units had a very strong tuning to the difference in rank between the two colors. Specifically, their activity increased or decreased monotonically with the difference in rank

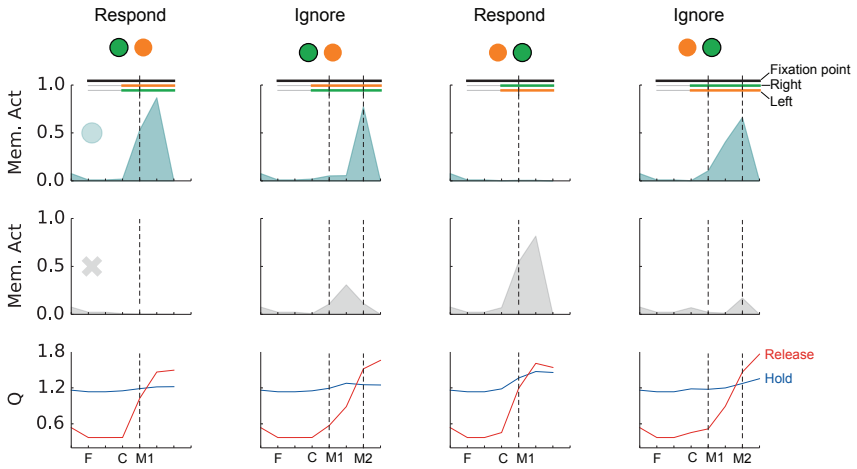


Figure 5.4: ***Example activity traces of a network trained on the full color-ranking task.*** *Top panels show the behavior of two memory units with task-relevant tuning. The top unit (light blue circle) responded most strongly to if the highest-ranking color was on the left (green had a higher rank than orange), and the middle unit (grey cross) prefers the highest rank on the right. The insets at the top show the time point when the fixation marker (F, black line) and the grey stimuli appeared on the screen, and when the colors turned on (C, bottom: left stimulus color, middle: right stimulus color). The black vertical line indicates the onset of a motion stimulus (M). In "respond" trials the first motion change occurred on the target side, and on "ignore" trials the first change occurred on the distracter side (M1) and then on the target side (M2). The bottom panels show action values that the model predicted for the "Hold" (blue) and "Release" (red) actions.*

between the two colors. In the next section, we provide a detailed analysis of rank-difference tuning in the trained networks.

Tuning to the difference in rank between colors

We found that the sensitivity of the model to the difference in the rank of the colors was mainly expressed in the synaptic weights from the On- and Off-cells in the input layer onto the memory units in the association layer. To assess the rank-difference tuning of these memory units in more detail we analyzed the synaptic weights from the input layer units that coded the colors on the left and right. Figure 5.5A shows the input connections of the two memory units of the network of Figure 5.4. There was a clear relation between synaptic weights and the rank and location of the two colored stimuli. The unit on the left of Figure 5.5A (light blue traces in

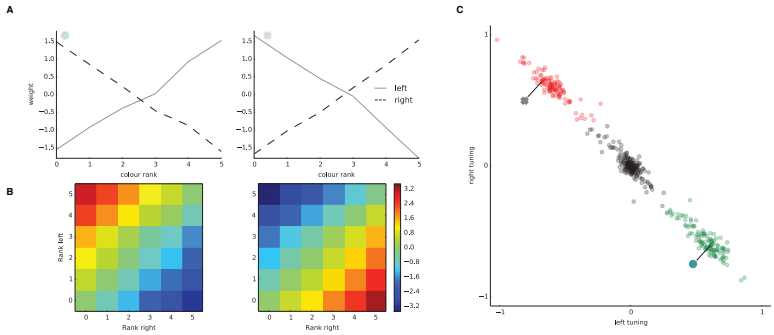


Figure 5.5: ***Tuning of memory units to the difference in rank between the two colors.*** **A**, *Visualization of weights from "on" sensory units to example memory units (same as those shown in Figure 5.5), ordered by rank. The solid line marks the synaptic weights from input units coding colors of different ranks in the left visual field, and the dashed line marks the weights coming from the right visual field.* **B**, *Sum of activations due to the presence of all possible combinations of color stimuli.* **C**, *Scatter-plot of linear-regression parameters (see main text). These results are based on synapses between the On-cells in the input layer and the memory units in the association layer. Off-cells and regular units usually did not have strong weights. The two example neurons from panel A (and Figure 5.4) have been marked with a light blue circle and a grey cross. Neurons that prefer stimuli on the right have been colored red, and neurons that prefer stimuli on the left are shown in green. Neurons that do not have strong opposite tuning for colors on the left and right are marked in black. Color labels were obtained by k-Means clustering (see Results).*

Figure 5.4) had positive weights to colors of increasing rank on the left and negative weights for colors of increasing rank on the right. The unit on the right (grey traces in Figure 5.4) had the opposite tuning and prefers stimuli with a higher rank on the right. The panels in Figure 5.5B show the amount of input to these two memory units for all color combinations, as determined by the weights in Figure 5.5A. The unit on the left of the figure received strong input if the higher ranked stimulus was on the left, and its activity depended on the distance in rank between the two stimuli as soon as the color cues were presented. The unit on the right of the figure had the opposite tuning to color combinations.

In order to quantify the prevalence of this gradual opposite left-right tuning to the color rank across units in the association layer of all trained networks, we computed linear regression coefficients ($a_{l/r}$) between color ranks ($R_{l/r}$) and synaptic weights w from the left and right (l/r) retinotopic

location:

$$w(R_l) = a_l R_l + b_l \,, \tag{5.25}$$

$$w(R_r) = a_r R_r + b_r \,, \tag{5.26}$$

Figure 5.5C shows the regression coefficients for left and right stimuli ($a_{l/r}$). It can be seen that many memory units exhibited a tuning similar to that of the two units illustrated in Figure 5.5A. Units with positive weights from stimuli with a high rank on one side almost invariably also had positive weights from stimuli with a low rank on the other side.

The memory units of the association layer fell into three groups: "left" (green in Figure 5.5C), "right" (red), and "non-tuned" (black) units. We labelled the units by performing k-Means clustering (assuming three clusters), and used this labeling to investigate how the networks solved the task. This analysis revealed that most networks had one unit preferring "left" stimuli (1.12 ± 0.35; mean ± s.e.m) and another one preferring "right" stimuli (1.10 ± 0.33), while the remaining units fell into the grey category of Figure 5.5C without clear tuning (1.78 ± 0.48).

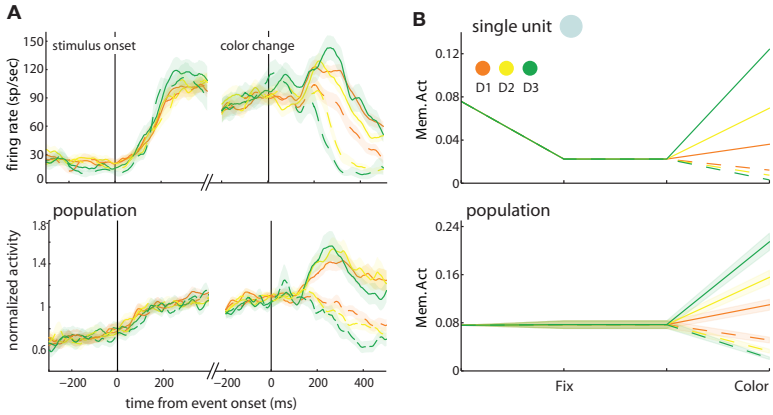


Figure 5.6: ***Rank difference coding.*** **A**, *Rank difference coding of a single cell in the prefrontal cortex of a monkey (top) and a population of prefrontal neurons (bottom, both adapted from Lennert and Martinez-Trujillo (2011)). Solid lines: target pattern on the preferred side, dashed lines: target on non-preferred side.* **B**, *Rank difference coding in a single model unit (top) and over the whole population of trained model units. Solid lines: target at preferred location, dashed lines: distracter at preferred location. Shadings show s.e.m.*

In monkeys, task-relevant neurons encode information about the rank-difference of stimuli and thereby which of the two stimuli should

be monitored for a motion change. One example neuron is shown in Figure 5.6A. This neuron increased its response if the target stimulus appeared on the preferred side of the neuron and in particular if the difference in color rank between the preferred and non-preferred was large. A similar effect was observed at the population level when cells were aligned according to their preferred side, which differed across neurons (Figure 5.6B, bottom). When we examined the activity of the top (light blue) "left" coding unit in response to color pairs with different rank-distances we found that the unit's activation after the onset of the color stimuli also varied monotonically with the difference in rank. Its activity was enhanced when the to-be-attended stimulus was on the left and suppressed when it was on the right, and this differential response increased with the difference in rank (Figure 5.6B, top). A similar pattern also held at the population level (Figure 5.6B, bottom), where we used the labels obtained by *k*-Means clustering (Figure 5.5C) to assign units to the left and right-coding groups. Thus, units in the networks that are trained with the AuGMEnT learning rule acquire a selectivity that resembles the tuning of neurons in the dorsolateral prefrontal cortex of monkeys.

In the next section we will investigate one important remaining question about the solution that was learned by these neural networks. How do networks deal with the motion stimuli, and in particular, how do they filter out stimuli on the distracter side, while monitoring stimuli on the target side as they instruct the model to release the button?

Response to the motion stimulus on the relevant and irrelevant side

To illustrate the attentional filtering mechanism, we will first focus our analysis on the example "left" coding unit that has been illustrated in Figures 5.4, 5.5A and 5.6B (indexed by a light blue circle). We recorded the unit's activation in response to the first motion stimulus for three color pairs of increasing distance. We compared two types of trials; "respond" trials where the target stimulus was on the left side and the motion changed occurred on the same side, and "ignore" trials where the target stimulus was on the right but the first motion change occurred on the left. These two trials types are of interest, because the motion stimulus occurs on the left, but in one case it is a target and in the other case it is a distracter, so we can specifically study the effect of attentional filtering. Figure 5.7 shows the influence of the left motion stimulus on the activity of the association unit when attended (left panel) versus when it needs to be ignored (right panel). When the left stimulus needs to be attended, the inputs from the color input

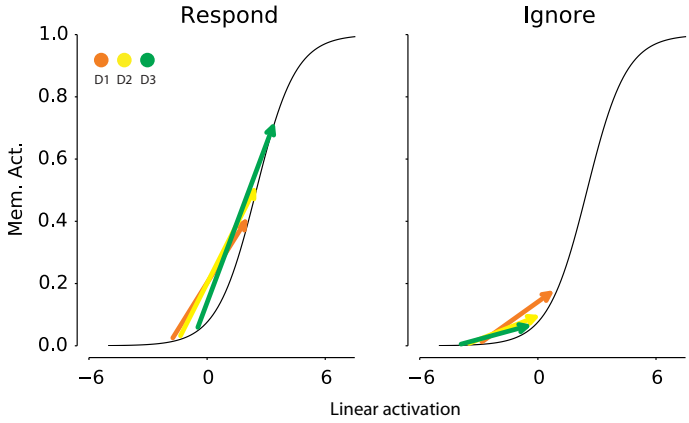


Figure 5.7: ***Attentional filtering by an example "left" unit.*** **Left**, *Color combination with the highest rank on the left brings the unit close to the steep part of the non-linear activation function (black sigmoidal curve). The appearance of the motion stimulus can therefore cause a large increase in the unit's activity. The three arrows show the increase in activity for color combinations with a distance in rank of* 1, 2 *and* 3. *This increase in activity is propagated to the output layer to cause an increase of the Q-value of the button release action.* **Right**, *Color combinations cueing that the right motion stimulus is relevant cause suppression so that the unit is relatively insensitive to a change in motion direction on the left side. Thus, a motion change on the irrelevant side cannot cause a strong increase in the Q-value of the release action and will be ignored by the model.*

units brought the activity close to the steep part of the unit's non-linear activation function so that the additional motion input causes a substantial increase in activity. This increased activity enhanced the *Q*-value of the release action because there was an excitatory connection from this memory unit onto the *Q*-value unit coding the release-action.

In contrast, when the left stimulus had to be ignored, the unit received less input from the color input units so that its activity was farther from the steep part of the activation function. Now the motion input caused a smaller increase in activity so that it is effectively ignored because it did not lead to a large increase of the *Q*-value of the release action. Most networks that we investigated employed this mechanism but we also found the inverse solution, where the activity of the memory unit was high and the motion stimulus on the relevant side inhibited the memory unit, which in turn disinhibited the 'release' action through an inhibitory connection. It is intriguing that this attentional filtering problem can be solved by an appropriate weighting of color inputs to memory units, without an

influence of feedback connections on the firing rate in sensory modules (which was absent in our model; see the next section). In our model, the feedback connections are only necessary for guiding plasticity. Our results obviously do not exclude that top-down effects on firing rates in sensory cortices are essential in other tasks (such as visual search). Our analysis thereby provides insight in how attentional filtering can be implemented by a simple feedforward neuronal network, and how it can be learned with a biologically plausible reinforcement learning scheme.

Learning of feedback connections

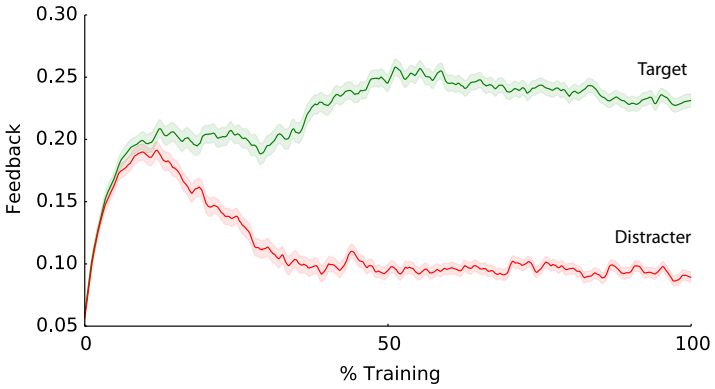


Figure 5.8: ***Learning shapes attentional feedback from the response selection stage.*** *Mean summed attentional feedback arriving at memory units encoding information about the target side (green) versus the distracter side (red) throughout learning. Shading shows s.e.m. We averaged feedback across all trial types. Because they did not occur equally often (due to the random generation of trial types), we estimated the relative position of trials of every type in the learning sequence with linear interpolation before averaging them.*

Finally, we investigated how the amount of top-down attention for the pattern on the left and right side evolve during learning. Although in the current model the feedback connections from the response selection stage to the association units are only used to gate plasticity and do not influence the activity of units at earlier processing levels, our main aim was to investigate how these feedback connections can be learned. In order to assess how feedback connections change during the course of learning, we measured the summed feedbacks arriving at memory units for the different trial types

throughout learning for the 100 trained networks. We compared the amount of feedback from the response selection stage (i.e. through the connections marked by dashed lines in Figure 5.1B) to "left" and "right" memory units (Figure 5.5C) in trials where attention had to be directed to the left and to the right and integrated the total amount of feedback that arrived during the trials (Figure 5.8). Specifically, for each trial of duration T that networks experienced, we computed the quantity $\sum_{0<t\leq T} f b_m^M(t)$ (methods) for each memory unit.

The effect of learning is clear—during initial learning, when weights are random, units tend to receive an equal and increasing amount of feedback regardless of the trial type. After about 10% of the training time, the strength of the feedback to the units coding for the distracter side started to decrease, whereas the amount of feedback to the target side increased slightly until the end of training. Thus, during the learning process, units that code information about the target receive more feedback from the response selection stage than units that code information about the distracter.

5.4 Discussion

In recent years important studies have started to document how rewards teach attention in human and non-human primate observers. When subjects receive a high reward for a particular stimulus, then this stimulus is likely to attract more attention at a later point in time (Anderson, Laurent, and Yantis, 2011; Chelazzi et al., 2013; Della Libera and Chelazzi, 2009; Hickey, Chelazzi, and Theeuwes, 2010; Raymond and O'Brien, 2009). As a general finding, stimuli that have been associated with a high reward evoke stronger neuronal responses than stimuli that have been associated with lower rewards in many brain structures including the motor cortex (Pastor-Bernier and Cisek, 2011), the parietal cortex (Peck et al., 2009; Platt and Glimcher, 1999), and visual cortex (Serences, 2008; Stănişor et al., 2013). In primary visual cortex, the neurons that are influenced by visual attention are also the ones that are influenced by reward expectancy (Stănişor et al., 2013), which suggests that there is a unified selection mechanism that is driven by reward expectancy as well as by shifts of attention (Maunsell, 2004). Such an effect of the reward contingency on the distribution of attention is expected because the contingency determines which information is task-relevant and which information can be ignored. In other words, there are strong theoretical grounds to believe that rewards

should indeed control attention. However, the precise mechanisms that explain how a reward in one trial can influence the deployment of attention in a later trial have not been well understood.

Here we have shown how a neural network trained with a biologically plausible reinforcement learning rule can learn an attentional control task when the only feedback from the environment is the occasional reward for correct performance. The performance of the networks trained with AuGMEnT exhibited a number of similarities with the performance of monkeys, and the activities of network units provide new insights into the changes in neuronal tuning that emerge during learning. First, the models exhibited a pattern of generalization to unseen color combinations that was remarkably similar to the generalization performance of the monkeys. Second, model units acquired a strong tuning to the difference in rank-order between the two color stimuli, just as was found in the prefrontal cortex of monkeys that had been trained on the task. Third, these newly formed representations explain why pairs of stimuli nearby in rank are associated with more errors than pairs of stimuli with larger differences in rank, because the activity of units in the association layer is more similar for stimuli with nearby ranks. Fourth, the simulation provided insight in how a neural network can decrease its sensitivity to stimuli that are task-irrelevant thereby filtering out distracting information. Finally, the present results illustrate how reward 'teaches' attention. The modified representations increased the amount of attentional feedback that was sent to units coding for the relevant motion stimulus, as instructed by the color cues.

The AuGMEnT learning rule uses two factors to gate neuronal plasticity, and their joint action at the synapse can provide a learning rule that is as powerful as the biologically implausible error-backpropagation rule (Roelfsema and van Ooyen, 2005). The first factor is a reward prediction error, which can be computed based on the difference in activity of the *Q*-value units that are selected in consecutive time-steps and has been included in many previous models on RL (e.g. Ashby, Ennis, and Spiering, 2007; Dayan and Yu, 2002; Sutton and Barto, 1998). This globally released signal informs all the synapses of the network whether the outcome of the previous action was better or worse than expected. Previous neurophysiological studies have demonstrated that many dopamine neurons in the substantial nigra and ventral tegmental area carry such reward-prediction errors (Schultz, 2002). These dopamine neurons have relatively widespread connections so that many synapses in the brain could pick up this reward prediction signal, although other neuromodulatory systems such as acetylcholine (Kilgard and Merzenich, 1998) or serotonin

(Liu et al., 2014) could play equivalent roles.

The second factor that gates learning is the attentional feedback from the response selection stage (Roelfsema and van Ooyen, 2005; Rombouts, Bohte, and Roelfsema, 2012, 2015). This feedback signal originates from the units that code for the action that was selected and it assigns credit to units at earlier processing levels that were responsible for this choice. The reciprocity of feedforward and feedback connections ensures that the units at the lower levels that provide the strongest input to the selected action are also the ones to receive a strong feedback signal. They are the ones to change their synaptic strengths (they will have the tag) as instructed by the reward prediction error (the globally released neuromodulator). A role of attentional feedback in the gating of learning is supported by studies demonstrating that subjects learn more readily about attended than non-attended features and objects (Ahissar and Hochstein, 1993; Jiang and Chun, 2001; Trabasso and Bower, 1968). Furthermore, studies in eye movement research have firmly established that attention is invariably directed to those items that are selected for a motor response (Deubel and Schneider, 1996; Kowler et al., 1995), in accordance with the proposed feedback scheme.

At first sight, our reasoning that the feedback connections gate learning and that they themselves are learned at the same time may seem circular. How can connections gate their own plasticity? The key observation is that the feedback pathways tag those synapses of the feedforward pathways that were responsible for the selected action. These tags are a prerequisite for synaptic change based on the globally released neuromodulator. If the action resulted in an outcome that was better than expected, the tagged synapses increase in strength to promote the future selection of the same action. If the outcome of the action was disappointing, then these synapses decrease in strength.

The resulting improvements in the feedforward pathways need to be accompanied by equivalent changes in the feedback pathways, as the reciprocity ensures that the credit in later trials will also assigned accurately, in spite of the modified feedforward connections. In the present work, the only influence of the attentional feedback signal is the deposit of synaptic tags for credit assignment and we did not model the well-established influence of feedback connections on the firing rates in lower level brain regions (Reynolds and Chelazzi, 2004; Roelfsema, 2006). Future models that include top-down effects on firing rates may further expand the capabilities of neural networks that are trained with AuGMEnT-like learning rules.

The present study provides new insights in how rewards can teach

attention. It is remarkable that a simple network that starts with a random connectivity can learn a relatively complex task where the rank of two color cues determines which of two stimuli needs to be monitored for a change in motion direction, by trial and error. Trial and error learning with the AuGMEnT learning rule is versatile, because the same network and learning rule can teach networks to perform different tasks, including ones that require storage of information in working memory, non-linear mappings of sensory stimuli onto motor responses and tasks that require the integration of stochastic sensory evidence for a decision (Rombouts, Bohte, and Roelfsema, 2012, 2015).

We illustrated how the reward-prediction errors of RL theory can also provide powerful learning rules for the shaping of attentional feedback connections. These new results thereby provide insight in how a perceptual system may learn to focus attention on those features that are important to solve a cognitive task. We hypothesize that similar mechanisms are at work when we learn in less constrained environments as is the case, for example, when learning to play tennis. We anticipate that future work will address the possible generalizations of AuGMEnT-like learning rules to situations that are even more challenging or that they will point out their limitations.

6 Learning neural resets

Abstract

Working memory is a key component of intelligence that the brain implements as persistent neural activations. How do persistent neurons learn to store information, and how can they be made to *forget* this information once it is no longer relevant? When animals learn episodic tasks, neurons in prefrontal cortex learn to represent task ends. We show that a biologically plausible neural network model equipped with persistent memory and a 'reset' action can learn to store and forget information at task ends by reinforcement learning. The new model has competitive performance compared to a variety of (biologically implausible) models. This chapter is based on Rombouts, Roelfsema, and Bohte (2014).

6.1 Introduction

Animals can learn very complex tasks based on simple reward and punishment schemes. Such tasks may require working memory (e.g. Gottlieb and Goldberg, 1999; Yang and Shadlen, 2007) where the correct sequence of actions depends on past information. It could be argued that most, if not all, tasks require some form of working memory, making the study of these types of tasks particularly relevant.

Successful models for learning working memory tasks employ a form of persistent memory, (e.g. Bakker, 2002; O'Reilly and Frank, 2006; Peshkin, Meuleau, and Kaelbling, 1999; Rombouts, Bohte, and Roelfsema, 2012; Todd, Niv, and Cohen, 2009). From neuroscience, there is ample experimental evidence that brains contain neurons that have persistent activations in working memory tasks (e.g. Gottlieb and Goldberg, 1999; Yang and Shadlen, 2007). Persistent memory was also found to be powerful for supervised training of Recurrent Neural Networks (RNNs) (Gers, Schmidhuber, and Cummins, 2000). However, models that employ persistent memory also need some mechanism for forgetting (Gers,

Schmidhuber, and Cummins, 2000). The reason for this is intuitive: Imagine performing a sequence of tasks where the information from a previous task may interfere with the execution of the current task (known as proactive interference). Clearing memory representations at task boundaries eliminates this problem.

The mathematical framework for modeling the learning of optimal sequences of actions from rewards and punishments is Reinforcement Learning (RL, Sutton and Barto (1998)). A large body of work has modeled (animal) learning in the RL framework. Most work has focussed on tasks that can be modeled as Markov Decision Problems, or MDPs. Working memory tasks fall in the class of Partially Observable MDPs, or POMDP tasks (Todd, Niv, and Cohen, 2009) as they require information presented at some previous time to make an optimal decision at a later point in time.

In the RL literature, learning episodic tasks involves a transition to a special terminal state and a subsequent reset of all dynamic parameters such as eligibility traces before starting a new trial (Sutton and Barto, 1998). However, autonomously learning agents do not have access to such 'trial-end' information. Learning task-ends should therefore be part of learning the task, and this may in fact not be trivial. When interacting with a new task, the animal has to learn that actions performed in the current task do not influence rewards in the next task, i.e. that they are independent. If tasks are independent, then it is beneficial to also clear working memory.

Experimental neuroscience shows that animals indeed learn to recognize the endings of tasks. For instance, in Fujii and Graybiel (2003), monkeys need to make a sequence of eye movements to visual targets. Neurons recorded in Prefrontal Cortex (PFC) show a burst of activation at the end of trials–a 'trial-end' signal. This behavior is very robust: the burst remains time-locked to the stimulus predicting trial-end under a wide range of manipulations. When the burst is fired, monkeys do indeed stop task execution. The authors suggest that the 'stop' signal could be used to reset PFC representations of the task held in working memory.

Here, we introduce *re*-AuGMEnT, a biologically plausible neural network model that can learn whole tasks, including trial ends, by reinforcement learning. The model is based on AuGMEnT (Rombouts, Bohte, and Roelfsema, 2012), which can learn challenging working memory tasks but requires supervised 'reset'-signals to reset working memory representations. We propose to include an 'internal' action that implements the 'reset'. This action can be included naturally in the AuGMEnT framework, and appropriately timed reset signals can then indeed be learned. As far as we are aware, this is the first work that

suggests a possible role for 'end' signals found in the brain (Fujii and Graybiel, 2003) within the RL framework. We show that the model can learn the tasks in Rombouts, Bohte, and Roelfsema (2012), and we show that AuGMEnT and *re*-AuGMEnT outperform a range of other RL working memory models on a T-Maze task.

This chapter is organized as follows. We first briefly introduce the original AuGMEnT framework of Rombouts, Bohte, and Roelfsema (2012), and detail how we include learnable resets. We then show how the model compares on a set of working memory tasks that are used in neuroscience. We further show how both models perform on a T-Maze task, and compare to a range of other RL working memory models (RL-LSTM, Memory Bits and Elman networks (Bakker, 2002; Peshkin, Meuleau, and Kaelbling, 1999)). We show that AuGMEnT significantly outperforms state-of-the art (RL-LSTM), and that *re*-AuGMEnT, *without being given resets*, is at least on par with Memory Bits and Elman networks which *do* have access to trial-end signals. Finally, we show how *re*-AuGMEnT relates to important earlier ideas and models cited above.

6.2 Model

We describe *re*-AuGMEnT adopting the same notation as Rombouts, Bohte, and Roelfsema (2012). The reader already familiar with AuGMEnT can skip ahead to the next paragraph which describes the reset mechanism. We suppress time indices t when expressions do not include information from previous time-steps. *re*-AuGMEnT is a three layer neural network that computes Q-values for alternative actions (Sutton and Barto, 1998), and where the hidden layer includes integrating units that learn to store task relevant information. (Fig 6.1a). The top layer contains two types of units: instantaneous and transient on(+)/off(-) units. Instantaneous units $x_i(t)$ encode sensory variables $s_i(t)$ at time step t, and on/off units code truncated temporal derivatives of these same sensory variables as:

$$x_i^+(t) = [s_i(t) - s_i(t-1)]_+ \; ; \qquad x_i^-(t) = [s_i(t-1) - s_i(t)]_+ \; ,$$

where $[x]_+ = x$ for $x > 0$ and 0 otherwise. The hidden layer also contains two kinds of units; regular units (superscripted with R) and memory units (superscripted with M). Regular units j receive inputs from all instantaneous units in the input layer via weights v_{ij}^R (with v_{0j}^R a bias weight). Their activation y_j^R is determined by a standard sigmoidal transformation of their inputs a_j^R:

$$y_j^R = \sigma(a_j^R, \theta) = 1/\left(1 + \exp\left(\theta - a_j^R\right)\right) \text{ with } a_j^R = \sum\nolimits_i v_{ij}^R x_i \,.$$

where θ shifts the squashing function. Memory units integrate input from the on/off units $x_l' = \{x_l^+, x_l^-\}$ through connections v_{lm}^M:

$$a_m^M(t) = a_m^M(t-1) + \sum\nolimits_l v_{lm}^M x_l'(t),$$

where activations y_m^M are computed as $y_m^M = \sigma(a_m^M, \theta)$. Memory units and regular hidden units project to output layer units k through, respectively, connections w_{mk}^M and w_{jk}^R (with w_{0k}^R a bias weight). The Q-value for action k in state s, $q_{s,k}$ (where s is implicitly defined by the hidden layer activations) is:

$$q_{s,k} = q_k = \sum\nolimits_j y_j^R w_{jk}^R + \sum\nolimits_m y_m^M w_{mk}^M \,.$$

Given the Q-values computed by the network, actions are selected using a max-Boltzmann Winner-Takes-All (WTA) mechanism (Wiering and Schmidhuber, 1997) with exploration rate ϵ, situated in an external controller. The network learns by minimizing SARSA Temporal Difference errors $\delta(t)$ by stochastic gradient descent (Rombouts, Bohte, and Roelfsema, 2012):

$$E(t) = \frac{1}{2}\delta(t)^2 = \frac{1}{2}\left(r(t) + \gamma q_{s',K'}(t) - q_{s,K}(t-1)\right)^2 \,,$$

where r is the reward received after executing the action K in state s at time $(t-1)$, $q_{s',K'}$ is the predicted value of the winning action K' for the next state s', and γ is the discount rate. Learning is implemented by an interaction of feedforward and feedback signals and a globally available neuromodulator like dopamine which represents $\delta(t)$. The activation of the winning output unit is set to 1, and the activations of the other output units are set to 0: $z_k = \delta_{kK}$, where δ_{kK} is the Kronecker delta function. The winning unit sends feedback through feedback connections w' (Fig 6.1a, dashed connections). The feedback interacts with the feedforward activations to form synaptic tags Tag_{xy} on each connection commensurate to the degree that this connection influenced the action selection. Connection strengths w_{xy} are modified as $\Delta w_{xy} = \beta\delta Tag_{xy}$, with β the learning rate. Synaptic tags are then updated as:

$$\Delta Tag^R_{jk} = (\lambda\gamma - 1)Tag^R_{jk} + y^R_j z_k \,,$$
$$\Delta Tag^M_{jk} = (\lambda\gamma - 1)Tag^M_{mk} + y^M_m z_k \,,$$
$$\Delta Tag^R_{ij} = (\lambda\gamma - 1)Tag^R_{ij} + w'^R_{Kj} y^R_j (1 - y^R_j) sTrace^R_{ij} \,,$$
$$\Delta Tag^M_{lm} = (\lambda\gamma - 1)Tag^M_{lm} + w'^M_{Km} y^M_m (1 - y^M_m) sTrace^M_{lm} \,,$$

with λ a decay parameter (Sutton and Barto, 1998) and w'^R_{Kj} and w'^M_{Kj} feedback weights from the output layer to the hidden layer. *sTraces* are intermediate variables:

$$sTrace^R_{ij} = x_i(t) \,; \qquad sTrace^M_{lm} = \sum\nolimits_{t'=0}^{t} x'_l(t') \,.$$

A *Tag* effectively encodes an eligibility trace (Sutton and Barto, 1998), and an *sTrace* encodes the history of activations that were transmitted through the associated synapse.

Resets When the 'reset' action is selected, parameters are updated as normal. Then, the values of memory *sTraces* and the activations of memory units are set to zero. After this reset, new feedforward activations are computed given the current observation, and an action is selected, with the restriction that the reset action cannot be selected twice in a row. Weights, *sTraces* and *Tags* are then again updated as defined above. In addition to these modifications to AuGMEnT (Rombouts, Bohte, and Roelfsema, 2012), we extended the observation layer with information about the reward and with the previous action of the network (including the new reset action). This is helpful as these signals can contain information about trial-ends.

6.3 Experiments

We tested our model on the same set of (non-linear) tasks as were used for testing AuGMEnT (Rombouts, Bohte, and Roelfsema, 2012) (Fig. 6.2b,c) and also to the T-Maze task from Bakker (2002) (Fig. 6.1d). For AuGMEnT we used networks with three regular units and four memory units in the hidden layer, as in Rombouts, Bohte, and Roelfsema (2012). As the *re*-AuGMEnT model expands the input layer, we used ten regular and ten memory units in the hidden layer. The number of output units (possible actions) depended on the task. For AuGMEnT we used the same parameters for the network

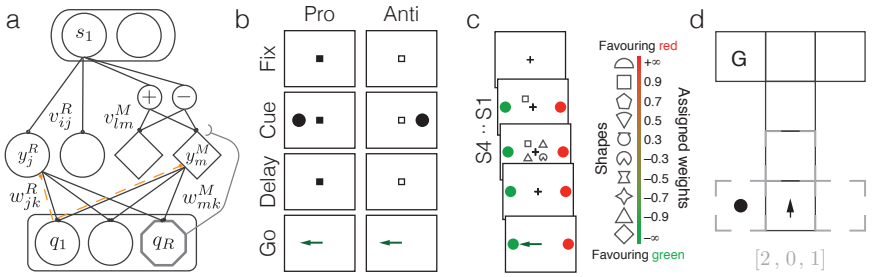


Figure 6.1: a, *AuGMEnT (black lines) and extended* re*-AuGMEnT (grey lines). Dashed lines: feedback connections, Diamonds: memory units* **b**, *Saccade/Antisaccade (SA) task after (Gottlieb and Goldberg, 1999).* **c**, *Probabilistic Classification (YS) task after (Yang and Shadlen, 2007).* **d**, *T-Maze task after (Bakker, 2002). Corridor length* $N = 3$*. Dashed lines show agent's observation space (coded as in grey numbers). Circle indicates location of reward at end of maze (G - invisible)*

and tasks as Rombouts, Bohte, and Roelfsema (2012). For *re*-AuGMEnT networks, we set $\beta = 0.05$ and $\lambda = 0.10$. Unless otherwise noted, results are based on runs with 100 randomly initialized networks.

Saccade/Antisaccade. The Saccade/Antisaccade (SA) task is based on (Gottlieb and Goldberg, 1999) (Fig. 6.1b). We implemented the task as in (Rombouts, Bohte, and Roelfsema, 2012). Briefly, the model is presented with a fixation mark (filled or empty square), then a cue (circle) appears on the left or right. After a delay the fixation mark turns off and the model has to select 'left' or 'right'. For the filled square the model has to select the cue direction and the opposite direction for the empty square. The trial ends without reward if the model breaks fixation before the fixation mark turns off. A correct trial yielded a reward of 1.5. We gave both models at most 1×10^5 trials to learn the task. After networks made 90% optimal choices for each of the four possible subtasks over the last 50 examples of each, we checked for correct performance by running 100 validation trials with β and ϵ set to 0.

Probabilistic Classification. The probabilistic classification task is based on (Yang and Shadlen, 2007) (Fig. 6.1c). Trial structure is similar to that of the SA task discussed above, but now four shapes are presented to the model. After a delay period with only the fixation mark visible, the model has to select 'left' or 'right'. The optimal choice depends on the four shapes s_1–s_4 that were shown; each shape has an associated weight (inset Fig. 6.1c) that

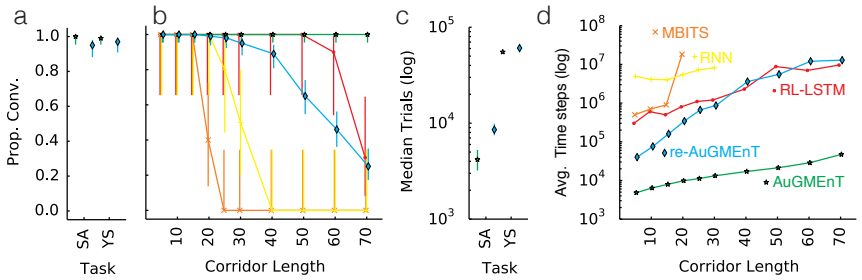


Figure 6.2: *Success rates for* **a**, *AuGMEnT (⋆) and* re*-AuGMEnT (◇) learning SA and YS task.* **b**, *T-Maze task ($N = 100$ for our models; $N = 10$ for results from (Bakker, 2002)). See labels inset in d. Bars:* 95% *confidence intervals (by* R *method* `prop.test`*).* **c**, *Median number of trials for learning the SA and YS tasks. Bars show 1st and 3rd quartile of distribution.* **d**, *Mean number of time-steps (*not *trials) for learning the T-Maze.*

determines the conditional reward distribution $P(Red|W) = 1/(1 + 10^{-W})$, with $W = \sum_{i=1}^{4} w(s_i)$; $P(Green|W) = 1 - P(R|W)$. The colored targets are randomly assigned to the left or right on each trial. The model received a reward of 1.5 for its choice for the Red or Green target according to the conditional distributions. We gave both models at most 5×10^5 trials to learn the task. Learning was complete when the model made 85% optimal choices over the last 2×10^4 trials.

T-Maze. The T-Maze task is based on Bakker (2002) (Fig. 6.1d). Information presented at the beginning of the maze is required to make optimal decisions at the end. The agent has actions `N,E,S,W` to move in all compass directions; task difficulty is scaled by increasing the corridor length N. When the agent remains in the same place (e.g. by moving into a wall), it receives a negative reward (−.1). The correct decision at the end of the maze is worth 4, and the wrong decision −.1. We added a time-out condition to the task: after $1.2N + 2$ time-steps we automatically stopped the trial, and started a new one. For the simulations we gave each network at most 5×10^5 trials to learn the task. Convergence was determined as for the SA task, but checked at 80% optimal choices as in Bakker (2002).

The results in Fig. 6.2a show that *re*-AuGMEnT is able to learn the same tasks as AuGMEnT with similar success rates. The learning process does take significantly longer, and the within-model variance of learning speed is also larger (Figure 6.2c). Given the increased complexity of the task, such increased learning time is to be expected. Figures 6.2b,d compare the

performance of AuGMEnT, *re*-AuGMEnT, and the methods tested in Bakker (2002): RL-LSTM, Memory Bits (Peshkin, Meuleau, and Kaelbling, 1999) and Elman Simple Recurrent Networks; note that none of these methods are biologically plausible. It is clear that AuGMEnT outperforms all other algorithms. At $N = 70$, this method still learns the task perfectly while convergence for RL-LSTM, the second best algorithm, drops to 30%. *re*-AuGMEnT does fairly well: it outperforms all models except RL-LSTM and AuGMEnT. Learning in *re*-AuGMEnT is significantly slower than the learning in AuGMEnT. This demonstrates that supervised reset signals contain a significant amount of information about the task. Importantly, both models automatically generalize over different delays (SA and YS tasks) and corridor lengths due to the on/off units; e.g. a *re*-AuGMEnT network that learned the $N = 5$ maze also solves the $N = 70$ maze (results not shown). This is not guaranteed for the other models.

6.4 Discussion

We demonstrated that it is possible to learn to recognize when working memory should be emptied in a biologically plausible neural network trained by a SARSA(λ) Temporal Difference learning rule. This is needed as, as soon as one requires persistent memory to solve problems, a form of "forgetting" is required to flush this working memory once it is no longer relevant (Gers, Schmidhuber, and Cummins, 2000). To achieve this, we started from a model that can learn working memory tasks when resets were provided in a supervised fashion (Rombouts, Bohte, and Roelfsema, 2012), and adapted this model so that it could learn when to reset its memory by trial and error. We first confirmed that the model could learn the tasks that can be learned by AuGMEnT using supervised resets; learning in *re*-AuGMEnT takes longer, which indicates that supervised resets provide significant information to an RL agent. We further compared AuGMEnT and *re*-AuGMEnT to the results for an RL version of the LSTM algorithm (RL-LSTM, Bakker (2002)), Memory Bits (Peshkin, Meuleau, and Kaelbling, 1999) and Elman networks on a T-Maze task. We found that AuGMEnT significantly outperforms all other algorithms on this task. The presented *re*-AuGMEnT model, which additionally has to learn to reset its memory, outperforms all algorithms except AuGMEnT and RL-LSTM.

The connection between memory and forgetting has been noted earlier: neural algorithms that include persistent memory are known to fail when this memory is not reset at appropriate times (Gers, Schmidhuber, and

Cummins, 2000). This was resolved for LSTM by coupling memory elements to special forget-gates that can reset the memory and the associated gradients in Gers, Schmidhuber, and Cummins (2000). These gates are sigmoidal units which set the 'leak' of an associated memory cell in a multiplicative fashion. The LSTM algorithm with forget gates is one of the most powerful supervised learning algorithms for training RNNs. While AuGMEnT shares a key idea of LSTM, we implemented resets by an internal reset action rather than forget-gates. A reset provides a binary on/off signal that is hard to achieve with sigmoidal gates, since gradients vanish at the extremes–solutions thus tend to be fit to typical timescales of the task, and do not automatically generalize to changes in delay length, unlike *re*-AuGMEnT. Some ideas in the RL literature are related to the current work. One notion is that of a memory cell that can be reset or overwritten by 'internal' actions as in Peshkin, Meuleau, and Kaelbling (1999) or Todd, Niv, and Cohen (2009), but neither of these models is biologically plausible. The PBWM model (O'Reilly and Frank, 2006), a biologically based learning scheme for learning working memory tasks is also closely related. Although the model is based in RL, it requires a teaching signal that provides the correct actions on each time-step and the architecture and learning rules are elaborate. In summary, we have shown that 'reset' actions allow a neural network to learn sequences of difficult working memory tasks, including when to forget, purely by trial-and-error learning. We hypothesize that such reset actions might explain the presence of task-end signals found in the brain (Fujii and Graybiel, 2003).

7 Summary

In this thesis we developed three models of reinforcement learning based on the framework of attention gated learning (Roelfsema and van Ooyen, 2005). In chapter 3, we reinterpreted the AGREL framework in terms of learning action-values. With this we showed how the framework can be applied to learning in settings with general rewards (compared to the $0-1$ rewards required in AGREL) and to settings where multiple simultaneous actions need to be selected (compared to the 1-of-c classification in AGREL) under the assumption of a scalar global prediction error signal. Preliminary work shows that the ideas in chapter 3 can be extended as to learn continuous actions with population coding output layers (Rombouts, Roelfsema, and Bohte, 2013). In chapter 4 we introduced AuGMEnT, which leaves the direct-reward setting of AGREL and MQ-AGREL for the delayed reward setting, thus showing how attention-gated learning can be used for learning action sequences, instead of single actions. The model is equipped with a working memory that can be used to learn difficult working memory tasks. The basic model can be seen as a biologically plausible implementation of the SARSA(λ) algorithm (Rummery and Niranjan, 1994), and the model with working memory extends this to learning problems in the class of partially observable MDPs (see chapter 2). The representations that the model learns are comparable to those that can be found in the brains of monkeys when they are trained on the same tasks. In chapter 5 we used AuGMEnT to model an attentional filtering task, investigating how such filtering can be learned by simple trial-and-error learning. One assumption we made for AuGMEnT is that a 'reset' signal is broadcast through the network after a trial ends, which can be seen as a supervised learning signal. In chapter 6 we extended AuGMEnT with an internal 'reset' action, and demonstrated that this extended model can learn trial boundaries, purely by reinforcement learning.

8 Samenvatting

Als je ooit een hond gehad hebt weet je dat je hem kan trainen door middel van beloning en straf. Hoe dit leren precies in zijn werk gaat wordt al sinds het begin van de twintigste eeuw bestudeerd. Een naam die wellicht bekend voorkomt is Ivan Pavlov, beroemd geworden met zijn conditioneringsexperiment. Als honden in hun natuurlijke omgeving gaan eten maken ze speeksel aan om met de vertering te helpen. In zijn beroemdste experiment liet Pavlov honden steeds een belletje horen, vlak voordat hij ze een stuk vlees gaf. Aan het begin van de training ging de hond pas kwijlen bij het verschijnen van het vlees, maar gestaag sloeg dit gedrag over naar het klinken van de bel. Dit liet zien dat de hond geleerd had om het verschijnen van vlees te voorspellen aan de hand van het geluid van de bel.

Een andere, minder bekende, wetenschapper is de Amerikaanse Edward Thorndike. Hij sloot katten op in 'puzzeldozen' waar ze uit moesten leren te ontsnappen voordat ze een beloning kregen. De kat moest bijvoorbeeld op een knop drukken om een ontsnappingsdeur te openen. Hij stelde vast dat het leren een regelmatig 'S' patroon volgde; eerst vertoonden de dieren willekeurige gedragingen totdat ze per ongeluk een keer op de knop drukten. Vervolgens leerden de dieren steeds sneller om op de knop te drukken, totdat ze een maximale snelheid behaalden. Aan de hand van zijn experimenten stelde Thorndike een aantal 'wetten' op, waaronder de 'wet van effect': iedere keer als een actie gevolgd wordt door een positieve uitkomst is het waarschijnlijker dat het dier dit gedrag zal vertonen, en iedere keer dat een actie gevolgd wordt door een negatieve uitkomst wordt het minder waarschijnlijk dat het dier dit gedrag zal vertonen. Het belangrijkste verschil met het werk van Pavlov is dat Thorndike's werk liet zien dat dieren in staat zijn om sequenties van acties te leren om beloning te krijgen, hetgeen meer op een natuurlijke situatie lijkt.

Dit proefschrift bestudeert het leren van beloning en straf, in het bijzonder hoe dit zou kunnen werken in de hersenen. Ik heb ideeën uit verschillende vakgebieden gebruikt om computermodellen te bouwen die, net als

dieren, kunnen leren van beloning en straf.

Allereerst heb ik gebruik gemaakt van ideeën uit het onderzoeksveld van 'Reinforcement Learning', ruw vertaald 'leren met bekrachtigingen'. Dit is een wiskundig raamwerk wat beschrijft dat agenten (met andere woorden, mensen en dieren, maar ook computerprogramma's) hun beloning willen maximaliseren in interactie met een complexe, onvoorspelbare wereld. Reinforcement Learning is gebaseerd op de observaties van onder andere Thorndike en Pavlov, maar ook door het wiskundige raamwerk van 'Dynamic Programming' uit de regeltechniek. Reinforcement Learning beschrijft niet alleen wat het doel van het leren is, maar ook hoe agenten dit doel kunnen bereiken. Het belangrijkste idee dat ik gebruikt heb is het idee van 'Temporal Difference Learning', ofwel TD-learning. Dit is een krachtig idee dat laat zien hoe een agent optimaal gedrag kan leren in een onbekende en onvoorspelbare wereld.

Het bijzondere is dat deze vorm van leren zeer direct gekoppeld blijkt aan wat bepaalde hersencellen (neuronen) doen terwijl dieren leren van beloning en straf. Deze cellen vormen dus een directe brug tussen theoretische modellen van optimaal leren en de biologie. Ze vormen ook een brug tussen gedrag dat we kunnen observeren aan de buitenkant (bijvoorbeeld kwijlen als de bel gaat) en wat er gebeurd in het complexe netwerk van neuronen wat we de hersenen noemen.

Dit brengt mij bij het laatste belangrijke ingrediënt; neurale netwerken. Neurale netwerken zijn abstracte, wiskundige beschrijvingen van de hersenen, en worden al sinds het midden van de 20ste eeuw bestudeerd. Interessant genoeg zijn deze neurale netwerken momenteel onze beste methode om computers taken te laten doen waar mensen goed in zijn maar computers niet, zoals bijvoorbeeld het herkennen van objecten in ingewikkelde omgevingen, of het begrijpen van spraak. De kern van neurale netwerken is het ontwikkelen van regels die beschrijven hoe koppelingen tussen neuronen moeten veranderen om bepaalde gedragingen te optimaliseren—dit worden leerregels genoemd.

In dit proefschrift heb ik leerregels ontwikkeld voor neurale netwerken die leren door beloning en straf. Deze leerregels zijn biologisch plausibel, omdat ze potentieel geïmplementeerd kunnen worden in hersenen, in tegenstelling tot andere leerregels die biologisch implausibel zijn, omdat ze bijvoorbeeld vereisen dat cellen toegang hebben tot niet locale informatie. De leerregels zijn gebaseerd op het AGREL 'Attention Gated Reinforcement Learning' model, ofwel aandachts-gestuurd leren. De voornaamste bijdrage van dit proefschrift is een generalisatie van AGREL naar het tijdsdomein, zodat het netwerk in staat is om sequenties van acties te leren om een belo-

ning te verkrijgen. Een ander belangrijk onderdeel is het toevoegen van een werkgeheugen wat het netwerk in staat stelt om taken te leren waar optimale acties niet direct gekoppeld zijn aan de huidige observatie, maar waar ze ook kunnen afhangen van eerdere observaties. Ik laat zien dat het model in staat is om ingewikkelde taken te leren, en dat het qua gedrag, zowel extern als intern lijkt op de gedragingen van echte dieren.

Appendix

Here we show that the AuGMEnT learning rules can be easily generalized to work for 'leaky' memory units, and that regular units and memory units can be considered as two extrema, instant-decay or no-decay, respectively.

To show this, we start from the activation-equations for memory units ((4.7) and (4.8)) reproduced here for convenience:

$$inp_m^M(t) = inp_m^M(t-1) + \sum_l v_{lm}^M x_l'(t)\,, \tag{8.1}$$

$$y_m^M(t) = \sigma(y_m^M(t))\,. \tag{8.2}$$

We can introduce a leak factor $\xi \in [0,1]$ that will introduce an exponential decay in the linear activation of the unit as follows[1]:

$$inp_m^M(t) = inp_m^M(t-1)\xi + \sum_l v_{lm}^M x_l'(t)\,. \tag{8.3}$$

where we obtain a regular unit when we set $\xi = 0$ and a memory unit when we set $\xi = 1$.

For arbitrary ξ, we need to make a small change in the derivation of the update rule for weights into the unit. By assuming that weights v_{lm}^M change only slowly, we can expand and rewrite $inp_m^M(t)$ as follows:

$$inp_m^M(t) = inp_m^M(t-1)\xi + \sum_l v_{lm}^M x_l'(t)\,, \tag{8.4}$$

$$\approx [\sum_l v_{lm}^M x_l'(t-1) + inp_m^M(t-2)\xi]\xi + \sum_l v_{lm}^M x_l'(t)\,, \tag{8.5}$$

$$\approx \sum_l v_{lm}^M [\sum_{t'=0}^{t} x_l'(t')\xi^{t-t'}]\,, \tag{8.6}$$

[1] It is also relatively straightforward to derive updates for time- and or input-dependent $\xi(t)$, following the same pattern.

If we determine the gradient $\frac{\partial inp_m^M(t)}{\partial v_{lm}^M}$, we thus obtain simply:

$$\frac{\partial inp_m^M(t)}{\partial v_{lm}^M} \approx [\sum_{t'=0}^{t} x_l'(t')\xi^{t-t'}] , \tag{8.7}$$

which looks similar to an sTrace, but with exponential discounting of previous inputs. It is easy to show that rewriting the sTrace update as:

$$\mathrm{d}sTrace_{lm} = (\xi - 1)sTrace_{lm}(t) + x_l'(t) , \tag{8.8}$$

results in the term required in the above gradient. These two small changes in the model suffice to generalize the AuGMEnT learning rules to a unified form, where regular units are obtained for $\xi = 0$ and memory units for $\xi = 1$.

It is interesting to consider the form of the synaptic trace update: its time-scale of decay must be equivalent to the time-scale of the (leaky) memory unit in order to estimate the correct gradient. By turning this reasoning around, we can argue that this coupling makes it unlikely that the memory as discussed in our model resides in the cell body, as it would be difficult to find the correct update when inputs have different decays. A model that has *synaptic* memory would be more plausible, with the added benefit that the 'activation' of the synapse is directly equivalent to the synaptic trace required for the update of the associated weight. This 'synaptic' memory can be seen as a prediction of AuGMEnT, and it would be interesting to see if this can be verified by experiment.

Bibliography

Ahissar, M. and Hochstein, S. (1993). "Attentional control of early perceptual learning." *Proceedings of the National Academy of Sciences of the United States of America* 90.12, pp. 5718–5722.

Anderson, B. A., Laurent, P. A., and Yantis, S. (2011). "Value-driven attentional capture." *Proceedings of the National Academy of Sciences of the United States of America* 108.25, pp. 10367–10371.

Anderson, B. A., Theeuwes, J., and Chelazzi, L. (2015). "Special Issue: Reward Guides Visual Attention: Selection, Learning and Motivation." *Visual Cognition* 2.1.

Ashby, F. G., Ennis, J. M., and Spiering, B. J. (2007). "A neurobiological theory of automaticity in perceptual categorization." *Psychological Review* 114.3, pp. 632–656.

Azevedo, F. A. C. et al. (2009). "Equal numbers of neuronal and nonneuronal cells make the human brain an isometrically scaled-up primate brain." *Journal of Comparative Neurology* 513.5, pp. 532–541.

Baird, L. (1995). "Residual algorithms: Reinforcement learning with function approximation." *Proceedings of the 26th International Conference on Machine Learning (ICML)*, pp. 30–37.

Bakker, B. (2002). "Reinforcement learning with long short-term memory." *Advances in Neural Information Processing Systems 15*.

Barak, O. et al. (2013). "From fixed points to chaos: three models of delayed discrimination." *Progress in Neurobiology* 103, pp. 214–222.

Barto, A. G., Sutton, R. S., and Brouwer, P. S. (1981). "Associative search network: A reinforcement learning associative memory." *Biological Cybernetics* 40.3, pp. 201–211.

Bellman, R. E. (1957). *Dynamic Programming*. Princeton Univ. Press.

Bishop, C. M. (1995). *Neural networks for pattern recognition*. Oxford Univ. Press.

Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). "A training algorithm for optimal margin classifiers." *Proceedings of the fifth annual workshop on Computational learning theory (COLT)*, pp. 144–152.

Botvinick, M. M., Niv, Y., and Barto, A. G. (2009). "Hierarchically organized behavior and its neural foundations: A reinforcement learning perspective." *Cognition* 113.3, pp. 262–280.

Boyan, J. and Moore, A. W. (1995). "Generalization in reinforcement learning: Safely approximating the value function." *Advances in Neural Information Processing Systems*, pp. 369–376.

Breland, K. and Breland, M. (1961). "The misbehavior of organisms." *American Psychologist* 16.11, p. 681.

Bromberg-Martin, E. S., Matsumoto, M., and Hikosaka, O. (2010a). "Distinct Tonic and Phasic Anticipatory Activity in Lateral Habenula and Dopamine Neurons." *Neuron* 67.1, pp. 144–155.

Bromberg-Martin, E. S., Matsumoto, M., and Hikosaka, O. (2010b). "Dopamine in Motivational Control: Rewarding, Aversive, and Alerting." *Neuron* 68.5, pp. 815–834.

Bundesen, C., Habekost, T., and Kyllingsbæk, S. (2005). "A Neural Theory of Visual Attention: Bridging Cognition and Neurophysiology." *Psychological Review* 112.2, pp. 291–328.

Cassenaer, S. and Laurent, G. (2012). "Conditional modulation of spike-timing-dependent plasticity for olfactory learning." *Nature* 482.7383, pp. 47–52.

Chang, Y. H., Ho, T., and Kaelbling, L. P. (2004). "All learning is local: Multi-agent learning in global reward games." *Advances in Neural Information Processing Systems* 17.

Chelazzi, L. et al. (2013). "Rewards teach visual selective attention." *Vision Research* 85, pp. 58–72.

Ciresan, D. C. et al. (2010). "Deep Big Simple Neural Nets Excel on Handwritten Digit Recognition." *Neural Computation* 22.12, pp. 3207–3220.

Corbetta, M. and Shulman, G. L. (2002). "Control of goal-directed and stimulus-driven attention in the brain." *Nature Reviews Neuroscience* 3.3, pp. 201–215.

Crick, F. (1989). "The recent excitement about neural networks." *Nature* 337.6203, p. 129.

Daw, N. D. (2012). "Model-based reinforcement learning as cognitive search: Neurocomputational theories." *Cognitive search: evolution algorithms and the brain*. The MIT Press, pp. 1–12.

Daw, N. D., Niv, Y., and Dayan, P. (2005). "Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control." *Nature Neuroscience* 8.12, pp. 1704–1711.

Dayan, P. and Balleine, B. W. (2002). "Reward, Motivation, and Reinforcement Learning." *Neuron* 38, pp. 285–298.

Dayan, P. and Yu, A. J. (2002). "ACh, uncertainty, and cortical inference." *Advances in Neural Information Processing Systems 15*.

De Pasquale, R. and Sherman, S. M. (2011). "Synaptic Properties of Corticocortical Connections between the Primary and Secondary Visual Cortical Areas in the Mouse." *Journal of Neuroscience* 31.46, pp. 16494–16506.

Deco, G., Rolls, E. T., and Romo, R. (2010). "Synaptic dynamics and decision making." *Proceedings of the National Academy of Sciences of the United States of America* 107.16, pp. 7545–7549.

Dehaene, S., Dehaene-Lambertz, G., and Cohen, L. (1998). "Abstract representations of numbers in the animal and human brain." *Trends in Neurosciences* 21.8, pp. 355–361.

Della Libera, C. and Chelazzi, L. (2009). "Learning to attend and to ignore is a matter of gains and losses." *Psychological Science* 20.6, pp. 778–784.

Desimone, R. and Duncan, J. (1995). "Neural mechanisms of selective visual attention." *Annual Review of Neuroscience* 18.1, pp. 193–222.

Deubel, H. and Schneider, W. X. (1996). "Saccade target selection and object recognition: Evidence for a common attentional mechanism." *Vision Research* 36.12, pp. 1827–1837.

Doshi-Velez, F. (2009). "The infinite partially observable Markov decision process." *Advances in Neural Information Processing Systems 22*, pp. 477–485.

Duff, M. O. (2002). "Optimal Learning: Computational Procedures for Bayes-Adaptive Markov Decision Processes." PhD thesis. University of Massachusetts Amherst.

Duncan, J. (2010). "The multiple-demand (MD) system of the primate brain: mental programs for intelligent behaviour." *Trends in Cognitive Sciences* 14.4, pp. 172–179.

Egorov, A. V. et al. (2002). "Graded persistent activity in entorhinal cortex neurons." *Nature* 420.6912, pp. 173–178.

Engel, T. A. and Wang, X.-J. (2011). "Same or Different? A Neural Circuit Mechanism of Similarity-Based Pattern Match Decision Making." *Journal of Neuroscience* 31.19, pp. 6982–6996.

Farley, B. G. and Clark, W. (1954). "Simulation of self-organizing systems by digital computer." *Information Theory, Transactions of the IRE Professional Group on* 4.4, pp. 76–84.

Felleman, D. J. and Van Essen, D. C. (1991). "Distributed hierarchical processing in the primate cerebral cortex." *Cerebral Cortex* 1.1, pp. 1–47.

Fiorillo, C. D. (2003). "Discrete Coding of Reward Probability and Uncertainty by Dopamine Neurons." *Science* 299.5614, pp. 1898–1902.

Fitzsimonds, R. M., Song, H.-j., and Poo, M.-m. (1997). "Propagation of activity-dependent synaptic depression in simple neural networks." *Nature* 368, pp. 439–448.

Fransén, E. et al. (2006). "Mechanism of Graded Persistent Cellular Activity of Entorhinal Cortex Layer V Neurons." *Neuron* 49.5, pp. 735–746.

Freedman, D. J. and Assad, J. A. (2006). "Experience-dependent representation of visual categories in parietal cortex." *Nature* 443.7107, pp. 85–88.

Freedman, D. J., Riesenhuber, M., et al. (2001). "Categorical representation of visual stimuli in the primate prefrontal cortex." *Science* 291.5502, pp. 312–316.

Frey, U. and Morris, R. G. M. (1997). "Synaptic tagging and long-term potentiation." *Nature* 385.6616, pp. 533–536.

Friedrich, J., Urbanczik, R., and Senn, W. (2011). "Spatio-Temporal Credit Assignment in Neuronal Population Learning." *PLoS Computational Biology* 7.6, e1002092.

Fujii, N. and Graybiel, A. M. (2003). "Representation of action sequence boundaries by macaque prefrontal cortical neurons." *Science* 301.5637, pp. 1246–1249.

Fukushima, K. (1980). "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position." *Biological Cybernetics* 36.4, pp. 193–202.

Funahashi, S., Bruce, C. J., and Goldman-Rakic, P. S. (1989). "Mnemonic coding of visual space in the monkey's dorsolateral prefrontal cortex." *Journal of Neurophysiology* 61.2, pp. 331–349.

Fusi, S., Drew, P. J., and Abbott, L. F. (2005). "Cascade Models of Synaptically Stored Memories." *Neuron* 45.4, pp. 599–611.

Gers, F. A., Schmidhuber, J., and Cummins, F. (2000). "Learning to forget: Continual prediction with LSTM." *Neural Computation* 12.10, pp. 2451–2471.

Gershman, S. J., Pesaran, B., and Daw, N. D. (2009). "Human Reinforcement Learning Subdivides Structured Action Spaces by Learning Effector-Specific Values." *Journal of Neuroscience* 29.43, pp. 13524–13531.

Gerstner, W. and Kistler, W. M. (2002). *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge University Press.

Gnadt, J. W. and Andersen, R. A. (1988). "Memory related motor planning activity in posterior parietal cortex of macaque." *Experimental Brain Research* 70.1, pp. 216–220.

Gold, J. I. and Shadlen, M. N. (2007). "The Neural Basis of Decision Making." *Annual Review of Neuroscience* 30.1, pp. 535–574.

Gottlieb, J. and Balan, P. (2010). "Attention as a decision in information space." *Trends in Cognitive Sciences* 14.6, pp. 240–248.
Gottlieb, J. and Goldberg, M. E. (1999). "Activity of neurons in the lateral intraparietal area of the monkey during an antisaccade task." *Nature Neuroscience* 2, pp. 906–912.
Gurney, K. N., Prescott, T. J., and Redgrave, P. (2001). "A computational model of action selection in the basal ganglia. I. A new functional anatomy." *Biological Cybernetics* 84.6, pp. 401–410.
Hernández, A. et al. (1997). "Discrimination in the sense of flutter: new psychophysical measurements in monkeys." *Journal of Neuroscience* 17.16, pp. 6391–6400.
Hickey, C., Chelazzi, L., and Theeuwes, J. (2010). "Reward Changes Salience in Human Vision via the Anterior Cingulate." *Journal of Neuroscience* 30.33, pp. 11096–11103.
Hikosaka, O. (2005). "Basal Ganglia Orient Eyes to Reward." *Journal of Neurophysiology* 95.2, pp. 567–584.
Hinton, G. E. (2007). "Learning multiple layers of representation." *Trends in Cognitive Sciences* 11.10, pp. 428–434.
Hinton, G. E. and Salakhutdinov, R. R. (2006). "Reducing the dimensionality of data with neural networks." *Science* 313.5786, pp. 504–507.
Hochreiter, S. and Schmidhuber, J. (1997). "Long short-term memory." *Neural Computation* 9.8, pp. 1735–1780.
Hodgkin, A. L. and Huxley, A. F. (1952). "A quantitative description of membrane current and its application to conduction and excitation in nerve." *The Journal of physiology* 117, pp. 500–544.
Hoerzer, G. M., Legenstein, R., and Maass, W. (2014). "Emergence of complex computational structures from chaotic neural networks through reward-modulated Hebbian learning." *Cerebral Cortex* 24.3, pp. 677–690.
Horvitz, J. C. (2000). "Mesolimbocortical and nigrostriatal dopamine responses to salient non-reward events." *Neuroscience* 96.4, pp. 651–656.
Houk, J. C., Adams, J. L., and Barto, A. G. (1995). "A model of how the basal ganglia generate and use neural signals that predict reinforcement." *Models of Information Processing in the Basal Ganglia*. Ed. by J. C. Houk, J. L. Davis, and D. G. Beiser. MIT Press, pp. 1–22.
Humphries, M. D., Stewart, R. D., and Gurney, K. N. (2006). "A Physiologically Plausible Model of Action Selection and Oscillatory Activity in the Basal Ganglia." *Journal of Neuroscience* 26.50, pp. 12921–12942.

Izhikevich, E. M. (2006). "Solving the Distal Reward Problem through Linkage of STDP and Dopamine Signaling." *Cerebral Cortex* 17.10, pp. 2443–2452.

Jaeger, H. (2001). "The" echo state" approach to analysing and training recurrent neural networks-with an erratum note'." *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report* 148.

Jiang, Y. and Chun, M. M. (2001). "Selective attention modulates implicit learning." *The Quarterly Journal of Experimental Psychology A* 54.4, pp. 1105–1124.

Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. (1996). *Planning and acting in partially observable stochastic domains*. Tech. rep. CS-96-08.

Kandel, E., Schwartz, J., and Jessell, T. (2013). *Principles of Neural Science, Fifth Edition*. McGraw Hill Professional.

Kilgard, M. P. and Merzenich, M. M. (1998). "Cortical Map Reorganization Enabled by Nucleus Basalis Activity." *Science* 279.5357, pp. 1714–1718.

Koulakov, A. A. et al. (2002). "Model for a robust neural integrator." *Nature Neuroscience* 5.8, pp. 775–782.

Kowler, E. et al. (1995). "The role of attention in the programming of saccades." *Vision Research* 35.13, pp. 1897–1916.

Krueger, K. A. and Dayan, P. (2009). "Flexible shaping: How learning in small steps helps." *Cognition* 110.3, pp. 380–394.

Le Cun, Y. (1988). "A theoretical framework for back-propagation." *Proceedings of the Connectionist Models Summer School*. Ed. by G. E. Hinton and T. J. Sejnowski. Citeseer, pp. 21–28.

Le, Q. V. et al. (2012). "Building high-level features using large scale unsupervised learning." *Proceedings of the 26th International Conference on Machine Learning (ICML)*.

Legenstein, R. et al. (2009). "Functional network reorganization in motor cortex can be explained by reward-modulated Hebbian learning." *Advances in Neural Information Processing Systems 22*, pp. 1105–1113.

Lennert, T. and Martinez-Trujillo, J. (2011). "Strength of Response Suppression to Distracter Stimuli Determines Attentional-Filtering Performance in Primate Prefrontal Neurons." *Neuron* 70.1, pp. 141–152.

Li, M. et al. (2014). "Efficient mini-batch training for stochastic optimization." *the 20th ACM SIGKDD international conference*. New York, New York, USA: ACM Press, pp. 661–670.

Liu, Z. et al. (2014). "Dorsal Raphe Neurons Signal Reward through 5-HT and Glutamate." *Neuron* 81.6, pp. 1360–1374.

Lo, C.-C. and Wang, X.-J. (2006). "Cortico–basal ganglia circuit mechanism for a decision threshold in reaction time tasks." *Nature Neuroscience* 9.7, pp. 956–963.

Louie, K., Grattan, L. E., and Glimcher, P. W. (2011). "Reward Value-Based Gain Control: Divisive Normalization in Parietal Cortex." *Journal of Neuroscience* 31.29, pp. 10627–10639.

Lovie, S. (2005). "History of Mathematical Learning Theory." *Encyclopedia of Statistics in Behavioral Science*. Ed. by B. Everitt and D. Howell. John Wiley & Sons, pp. 861–864.

Luk, C. H. and Wallis, J. D. (2009). "Dynamic Encoding of Responses and Outcomes by Neurons in Medial Prefrontal Cortex." *Journal of Neuroscience* 29.23, pp. 7526–7539.

Luo, S. X., Axel, R., and Abbott, L. F. (2010). "Generating sparse and selective third-order responses in the olfactory system of the fly." *Proceedings of the National Academy of Sciences of the United States of America* 107.23, pp. 10713–10718.

Maass, W., Natschläger, T., and Markram, H. (2002). "Real-time computing without stable states: A new framework for neural computation based on perturbations." *Neural Computation* 14.11, pp. 2531–2560.

Machens, C. K. (2005). "Flexible Control of Mutual Inhibition: A Neural Model of Two-Interval Discrimination." *Science* 307.5712, pp. 1121–1124.

Mao, T. et al. (2011). "Long-Range Neuronal Circuits Underlying the Interaction between Sensory and Motor Cortex." *Neuron* 72.1, pp. 111–123.

Matsumoto, K. (2003). "Neuronal Correlates of Goal-Based Motor Selection in the Prefrontal Cortex." *Science* 301.5630, pp. 229–232.

Maunsell, J. H. R. (2004). "Neuronal representations of cognitive state: reward or attention?" *Trends in Cognitive Sciences* 8.6, pp. 261–265.

McCulloch, W. S. and Pitts, W. (1943). "A logical calculus of the ideas immanent in nervous activity." *The bulletin of mathematical biophysics* 5.4, pp. 115–133.

Miller, E. K. and Cohen, J. D. (2001). "An integrative theory of prefrontal cortex function." *Annual Review of Neuroscience* 24.1, pp. 167–202.

Miller, P. and Wang, X.-J. (2006). "Inhibitory control by an integral feedback signal in prefrontal cortex: A model of discrimination between sequential stimuli." *Proceedings of the National Academy of Sciences* 103.1, pp. 201–206.

Minsky, M. (1954). "Theory of neural-analog reinforcement systems and its application to the brain-model problem." PhD thesis. Princeton University.

Minsky, M. and Papert, S. (1969). *Perceptrons*. The MIT Press.
Mnih, V. et al. (2013). "Playing Atari with Deep Reinforcement Learning." *arXiv.org*. arXiv: 1312.5602v1 [cs.LG].
Moncada, D. et al. (2011). "Identification of transmitter systems and learning tag molecules involved in behavioral tagging during memory formation." *Proceedings of the National Academy of Sciences of the United States of America* 108.31, pp. 12931–12936.
Montague, P. R., Dayan, P., and Sejnowski, T. J. (1996). "A framework for mesencephalic dopamine systems based on predictive Hebbian learning." *Journal of Neuroscience* 16.5, pp. 1936–1947.
Montague, P. R., Hyman, S. E., and Cohen, J. D. (2004). "Computational roles for dopamine in behavioural control." *Nature* 431.7010, pp. 760–767.
Moore, T. and Armstrong, K. M. (2003). "Selective gating of visual signals by microstimulation of frontal cortex." *Nature* 421.6921, pp. 370–373.
Morris, G. et al. (2006). "Midbrain dopamine neurons encode decisions for future action." *Nature Neuroscience* 9.8, pp. 1057–1063.
Nassi, J. J. and Callaway, E. M. (2009). "Parallel processing strategies of the primate visual system." *Nature Reviews Neuroscience* 10.5, pp. 360–372.
Niekum, S. et al. (2013). "Incremental Semantically Grounded Learning from Demonstration." *Robotics: Science and Systems*.
Niv, Y. (2009). "Reinforcement learning in the brain." *The Journal of Mathematical Psychology* 53.3, pp. 139–154.
O'Reilly, R. C. (1996). "Biologically plausible error-driven learning using local activation differences: The generalized recirculation algorithm." *Neural Computation* 8.5, pp. 895–938.
O'Reilly, R. C. and Frank, M. J. (2006). "Making working memory work: a computational model of learning in the prefrontal cortex and basal ganglia." *Neural Computation* 18.2, pp. 283–328.
O'Reilly, R. C., Hazy, T. E., and Herd, S. A. (2012). "The leabra cognitive architecture: how to play 20 principles with nature and win!" *The Oxford Handbook of Cognitive Science*.
O'Reilly, R. C. and Munakata, Y. (2000). *Computational Explorations in Cognitive Neuroscience: Understanding the Mind by Simulating the Brain*. the MIT Press.
Padoa-Schioppa, C. and Assad, J. A. (2006). "Neurons in the orbitofrontal cortex encode economic value." *Nature* 441.7090, pp. 223–226.
Parisien, C., Anderson, C. H., and Eliasmith, C. (2008). "Solving the problem of negative synaptic weights in cortical models." *Neural Computation* 20.6, pp. 1473–1494.

Pastor-Bernier, A. and Cisek, P. (2011). "Neural Correlates of Biased Competition in Premotor Cortex." *Journal of Neuroscience* 31.19, pp. 7083–7088.

Pavlov, I. P. (1928). "Experimental psychology and psycho-pathology in animals." *International Congress of Medicine* 14.

Peck, C. J. et al. (2009). "Reward Modulates Attention Independently of Action Value in Posterior Parietal Cortex." *Journal of Neuroscience* 29.36, pp. 11182–11191.

Peshkin, L., Meuleau, N., and Kaelbling, L. (1999). "Learning Policies with External Memory." *Proceedings of the 26th International Conference on Machine Learning (ICML)*.

Pineda, F. J. (1989). "Recurrent backpropagation and the dynamical approach to adaptive neural computation." *Neural Computation* 1.2, pp. 161–172.

Platt, M. L. and Glimcher, P. W. (1999). "Neural correlates of decision variables in parietal cortex." *Nature* 400.6741, pp. 233–238.

Potjans, W., Diesmann, M., and Morrison, A. (2011). "An Imperfect Dopaminergic Error Signal Can Drive Temporal-Difference Learning." *PLoS Computational Biology* 7.5, e1001133.

Potjans, W., Morrison, A., and Diesmann, M. (2009). "A spiking neural network model of an actor-critic learning agent." *Neural Computation* 21.2, pp. 301–339.

Raymond, J. E. and O'Brien, J. L. (2009). "Selective Visual Attention and Motivation The Consequences of Value Learning in an Attentional Blink Task." *Psychological Science* 20.8, pp. 981–988.

Redgrave, P. and Gurney, K. N. (2006). "The short-latency dopamine signal: a role in discovering novel actions?" *Nature Reviews Neuroscience* 7.12, pp. 967–975.

Redgrave, P., Prescott, T. J., and Gurney, K. N. (1999). "Is the short-latency dopamine response too short to signal reward error?" *Trends in Neurosciences* 22.4, pp. 146–151.

Reynolds, J. H. and Chelazzi, L. (2004). "Attentional modulation of visual processing." *Annual Review of Neuroscience* 27, pp. 611–647.

Richardson, R. T. and DeLong, M. R. (1986). "Nucleus basalis of Meynert neuronal activity during a delayed response task in monkey." *Brain research* 399.2, pp. 364–368.

Rieke, F. (1999). *Spikes: exploring the neural code*. The MIT Press.

Rigotti, M. et al. (2013). "The importance of mixed selectivity in complex cognitive tasks." *Nature* 497.7451, pp. 585–590.

Ring, M., Schaul, T., and Schmidhuber, J. (2011). "The two-dimensional organization of behavior." *IEEE International Conference on Development and Learning (ICDL)* 2, pp. 1–8.

Roelfsema, P. R. (2006). "Cortical algorithms for perceptual grouping." *Annual Review of Neuroscience* 29, pp. 203–227.

Roelfsema, P. R. and van Ooyen, A. (2005). "Attention-gated reinforcement learning of internal representations for classification." *Neural Computation* 17.10, pp. 2176–2214.

Roelfsema, P. R., van Ooyen, A., and Watanabe, T. (2010). "Perceptual learning rules based on reinforcers and attention." *Trends in Cognitive Sciences* 14.2, pp. 64–71.

Rombouts, J. O., Bohte, S. M., Martinez-Trujillo, J., et al. (2015). "A learning rule that explains how rewards teach attention." *Visual Cognition* 23.1-2, pp. 179–205.

Rombouts, J. O., Bohte, S. M., and Roelfsema, P. R. (2012). "Neurally Plausible Reinforcement Learning of Working Memory Tasks." *Advances in Neural Information Processing Systems*, pp. 1880–1888.

Rombouts, J. O., Bohte, S. M., and Roelfsema, P. R. (2015). "How Attention Can Create Synaptic Tags for the Learning of Working Memories in Sequential Tasks." *PLoS Computational Biology* 11.3, e1004060.

Rombouts, J. O., Roelfsema, P. R., and Bohte, S. M. (2013). "Biologically plausible reinforcement learning of continuous actions." *BMC Neuroscience*.

Rombouts, J. O., Roelfsema, P. R., and Bohte, S. M. (2014). "Learning Resets of Neural Working Memory." *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*.

Rombouts, J. O., van Ooyen, A., et al. (2012). "Biologically Plausible Multi-dimensional Reinforcement Learning in Neural Networks." *Artificial Neural Networks-ICANN*, pp. 443–450.

Romo, R., Brody, C. D., et al. (1999). "Neuronal correlates of parametric working memory in the prefrontal cortex." *Nature* 399.6735, pp. 470–473.

Romo, R., Hernández, A., and Zainos, A. (2004). "Neuronal correlates of a perceptual decision in ventral premotor cortex." *Neuron* 41.1, pp. 165–173.

Romo, R., Hernández, A., Zainos, A., and Salinas, E. (2003). "Correlated neuronal discharges that increase coding efficiency during perceptual discrimination." *Neuron* 38.4, pp. 649–657.

Romo, R. and Salinas, E. (2003). "Flutter Discrimination: neural codes, perception, memory and decision making." *Nature Reviews Neuroscience* 4.3, pp. 203–218.

Rosenblatt, F. (1962). *Principles of Neurodynamics*. Spartan Book.

Ross, S., Chaib-draa, B., and Pineau, J. (2007). "Bayes-Adaptive POMDPs." *Advances in Neural Information Processing Systems 20*, pp. 1225–1232.

Rothkopf, C. A. and Ballard, D. H. (2010). "Credit Assignment in Multiple Goal Embodied Visuomotor Behavior." *Frontiers in Psychology* 1.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). "Learning representations by back-propagating errors." *Nature* 323.6088, pp. 533–536.

Rummery, G. A. and Niranjan, M. (1994). *On-line Q-learning using connectionist systems*. Tech. rep. CUED/F-INFENG/TR 166. Cambridge.

Sajikumar, S. and Korte, M. (2011). "Metaplasticity governs compartmentalization of synaptic tagging and capture through brain-derived neurotrophic factor (BDNF) and protein kinase Mζ (PKMζ)." *Proceedings of the National Academy of Sciences of the United States of America* 108.6, pp. 2551–2556.

Samejima, K. et al. (2005). "Representation of Action-Specific Reward Values in the Striatum." *Science* 310.5752, pp. 1337–1340.

Satoh, T. et al. (2003). "Correlated coding of motivation and outcome of decision by dopamine neurons." *Journal of Neuroscience* 23.30, pp. 9913–9923.

Schoups, A. et al. (2001). "Practising orientation identification improves orientation coding in V1 neurons." *Nature* 412.6846, pp. 549–553.

Schultz, W. (2002). "Getting formal with dopamine and reward." *Neuron* 36.2, pp. 241–263.

Schultz, W. (2007). "Multiple Dopamine Functions at Different Time Courses." *Annual Review of Neuroscience* 30.1, pp. 259–288.

Schultz, W., Apicella, P., and Ljungberg, T. (1993). "Responses of monkey dopamine neurons to reward and conditioned stimuli during successive steps of learning a delayed response task." *Journal of Neuroscience* 13.3, pp. 900–913.

Schultz, W., Dayan, P., and Montague, P. R. (1997). "A neural substrate of prediction and reward." *Science* 275.5306, pp. 1593–1599.

Self, M. W. et al. (2012). "Different glutamate receptors convey feedforward and recurrent processing in macaque V1." *Proceedings of the National Academy of Sciences of the United States of America* 109.27, pp. 11031–11036.

Serences, J. T. (2008). "Value-Based Modulations in Human Visual Cortex." *Neuron* 60.6, pp. 1169–1181.
Serre, T. et al. (2007). "A quantitative theory of immediate visual recognition." *Progress in brain research* 165, pp. 33–56.
Seung, H. S. (2003). "Learning in spiking neural networks by reinforcement of stochastic synaptic transmission." *Neuron* 40.6, pp. 1063–1073.
Sherman, S. M. and Guillery, R. W. (1998). "On the actions that one nerve cell can have on another: distinguishing "drivers" from "modulators"." *Proceedings of the National Academy of Sciences* 95.12, pp. 7121–7126.
Singh, S. et al. (2000). "Convergence Results for Single-Step On-Policy Reinforcement-Learning Algorithms." *Machine Learning* 39, pp. 287–308.
Soltani, A. and Wang, X.-J. (2009). "Synaptic computation underlying probabilistic inference." *Nature Neuroscience* 13.1, pp. 112–119.
Sommer, M. A. and Wurtz, R. H. (2001). "Frontal eye field sends delay activity related to movement, memory, and vision to the superior colliculus." *Journal of Neurophysiology* 85.4, pp. 1673–1685.
Stănişor, L. et al. (2013). "A unified selection signal for attention and reward in primary visual cortex." *Proceedings of the National Academy of Sciences* 110.22, pp. 9136–9141.
Stewart, T. C., Bekolay, T., and Eliasmith, C. (2012). "Learning to select actions with spiking neurons in the Basal Ganglia." *Frontiers in neuroscience* 6.
Stuart, G. et al. (1997). "Action potential initiation and backpropagation in neurons of the mammalian CNS." *Trends in Neurosciences* 20.3, pp. 125–131.
Suri, R. E. and Schultz, W. (1998). "Learning of sequential movements by neural network model with dopamine-like reinforcement signal." *Experimental Brain Research* 121.3, pp. 350–354.
Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: an introduction*. MIT Press.
Sutton, R. S., Precup, D., and Singh, S. P. (1999). "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning." *Artificial Intelligence* 112.1, pp. 181–211.
Thorndike, E. (1898). "Some experiments on animal intelligence." *Science* 7.181, pp. 818–824.
Todd, M. T., Niv, Y., and Cohen, J. D. (2009). "Learning to use working memory in partially observable environments through dopaminergic reinforcement." *Advances in Neural Information Processing Systems* 21, pp. 1689–1696.

Trabasso, T. and Bower, G. H. (1968). *Attention in learning: Theory and research*. Krieger.

Treue, S. and Maunsell, J. H. (1996). "Attentional modulation of visual motion processing in cortical areas MT and MST." *Nature* 382.6591, pp. 539–541.

Urbanczik, R. and Senn, W. (2009). "Reinforcement learning in populations of spiking neurons." *Nature Neuroscience* 12.3, pp. 250–252.

Usher, M. and McClelland, J. L. (2001). "The time course of perceptual choice: the leaky, competing accumulator model." *Psychological Review* 108.3, pp. 550–592.

Van Hasselt, H. (2011). "Insights in Reinforcement Learning." PhD thesis. Utrecht University.

Vijayraghavan, S. et al. (2007). "Inverted-U dopamine D1 receptor actions on prefrontal neurons engaged in working memory." *Nature Neuroscience* 10.3, pp. 376–384.

Wallis, J. D. (2007). "Orbitofrontal Cortex and Its Contribution to Decision-Making." *Annual Review of Neuroscience* 30.1, pp. 31–56.

Watkins, C. J. C. H. and Dayan, P. (1992). "Q-learning." *Machine Learning* 8.3, pp. 279–292.

Whitehead, S. D. and Ballard, D. H. (1991). "Learning to perceive and act by trial and error." *Machine Learning* 7.1, pp. 45–83.

Widrow, B. and Hoff, M. E. (1960). "Adaptive switching circuits." *IRE WESCON Convention Record, Part 4*. New York: Defense Technical Information Center, pp. 96–104.

Wiering, M. and Schmidhuber, J. (1997). "HQ-learning." *Adaptive Behavior* 6.2, pp. 219–246.

Williams, G. V. and Goldman-Rakic, P. S. (1995). "Modulation of memory fields by dopamine D1 receptors in prefrontal cortex." *Nature* 376.6541, pp. 572–575.

Williams, R. J. (1992). "Simple statistical gradient-following algorithms for connectionist reinforcement learning." *Machine Learning* 8.3, pp. 229–256.

Wilson, D. R. and Martinez, T. R. (2003). "The general inefficiency of batch training for gradient descent learning." *Neural Networks* 16.10, pp. 1429–1451.

Wise, R. A. (2004). "Dopamine, learning and motivation." *Nature Reviews Neuroscience* 5.6, pp. 483–494.

Wolpert, D. M., Ghahramani, Z., and Jordan, M. I. (1995). "An internal model for sensorimotor integration." *Science* 269, pp. 1880–1882.

Yang, T. and Shadlen, M. N. (2007). "Probabilistic reasoning by neurons." *Nature* 447.7148, pp. 1075–1080.
Yu, A. J. and Dayan, P. (2005). "Uncertainty, Neuromodulation, and Attention." *Neuron* 46.4, pp. 681–692.
Zipser, D. (1991). "Recurrent network model of the neural mechanism of short-term active memory." *Neural Computation* 3.2, pp. 179–193.
Zipser, D. and Rumelhart, D. E. (1993). "The neurobiological significance of the new learning models." *Computational neuroscience*. Ed. by E. L. Schwartz. MIT Press, pp. 192–200.