ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

Learning Resets of Neural Working Memory

J.O. Rombouts[1], P.R. Roelfsema[2] and S.M. Bohte[1] *

1- Centrum Wiskunde & Informatica - Life Sciences
Science Park 123 - The Netherlands

2- Netherlands Institute for Neuroscience - Vision and Cognition
Meibergdreef 47 - The Netherlands

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.

1 Introduction

Animals can learn very complex tasks based on simple reward and punishment schemes. Such tasks may require working memory (e.g. [1, 2]) 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. [3, 4, 5, 6, 7]. From neuroscience, there is ample experimental evidence that brains contain neurons that have persistent activations in working memory tasks [1, 2]. Persistent memory was also found to be powerful for supervised training of Recurrent Neural Networks (RNNs) [8]. However, models that employ persistent memory also need some mechanism for forgetting [8]. 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, [9]). 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 [6] 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

*JOR is funded by NWO grant 612.066.826

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

traces before starting a new trial [9]. 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 beginnings and endings of tasks (e.g. [10]).

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 [7], 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 paper that suggests a possible role for 'end' signals found in the brain [10] within the RL framework. We show that the model can learn the tasks in [7], and we show that AuGMEnT and *re*-AuGMEnT outperform a range of other RL working memory models on a T-Maze task.

2 Model

We describe *re*-AuGMEnT adopting the same notation as [7] and indicating where the models differ. 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 [9], and where the hidden layer includes integrating units that learn to store task relevant information. (Fig 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:

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

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

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

w^M_{mk} and w^R_{jk} (with w^R_{0k} 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:

$$q_{s,k} = q_k = \sum\nolimits_j y^R_j w^R_{jk} + \sum\nolimits_m y^M_m w^M_{mk} \ .$$

Given the Q-values computed by the network, actions are selected using a max-Boltzmann Winner-Takes-All (WTA) mechanism [11] with exploration rate ϵ, situated in an external controller. The network learns by minimizing SARSA Temporal Difference errors $\delta(t)$ by stochastic gradient descent [7]:

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

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 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:

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

with λ a decay parameter [9] and ${w'}^R_{Kj}$ and ${w'}^M_{Kj}$ feedback weights from the output layer to the hidden layer. *sTraces* are intermediate variables:

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

The *sTrace* variables are new to *re*-AuGMEnT. A *Tag* effectively encodes an eligibility trace [9], and an *sTrace* encodes the history of activations that were transmitted through the associated synapse. Having separate *Tags* and *sTraces* has two advantages: all *Tags* are now treated identically (in contrast to [7] where $\lambda = 0$ for memory *Tags*) and *sTraces* can be reset independently from *Tags*.

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 [7], 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.

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

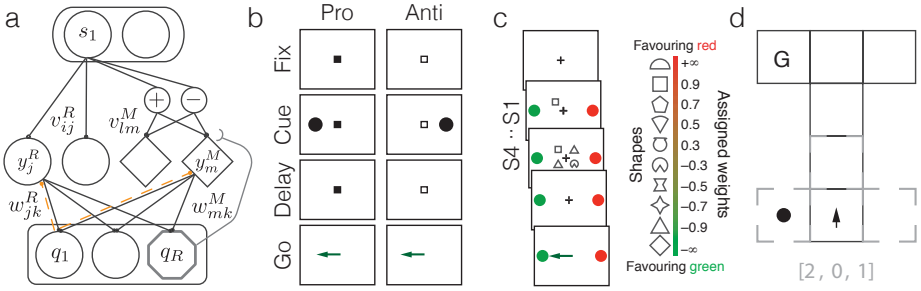

Fig. 1: (**a**) AuGMEnT (black lines) and extended *re*-AuGMEnT (grey lines). Dashed lines: feedback connections, Diamonds: memory units (**b**). Saccade/Antisaccade (SA) task after [1]. (**c**) Probabilistic Classification (YS) task after [2]. (**d**) T-Maze task after [4]. Corridor length $N = 3$. Dashed lines show agent's observation space (coded as in grey numbers). Circle indicates location of reward at end of maze (G - invisible)

3 Experiments

We tested our model on the same set of (non-linear) tasks as were used for testing AuGMEnT [7] (Fig. 2b,c) and also to the T-Maze task from [4] (Fig. 1d). For AuGMEnT we used networks with three regular units and four memory units in the hidden layer, as in [7]. 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 and tasks as [7]. 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 [1] (Fig. 1b). We implemented the task as in [7]. 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 [2] (Fig. 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. 1c) that determines the conditional reward distribution $P(Red|W) = 1/(1+10^{-W})$, with $W = \sum_{i=1}^{4} w(s_i)$; $P(Green|W) =$

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

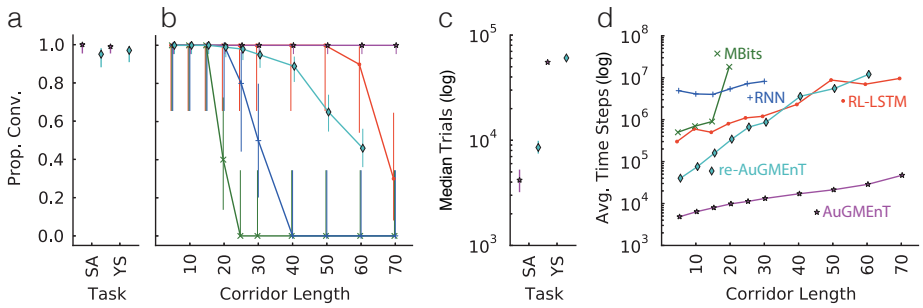

Fig. 2: Success rates for (**a**) AuGMEnT ($\star$) and *re*-AuGMEnT ($\diamond$) learning SA and YS task. (**b**) T-Maze task (N=100 for our models; N=10 for results from [4]). See labels inset in d. Bars: 95% confidence intervals (by `R` method `prop.test`). (**c**) Median number of trials for learning 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.

$1 - P(R|W)$. 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 [4] (Fig. 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 [4].

The results in Fig. 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 (Fig. 2c). Given the increased complexity of the task, such increased learning time is to be expected. Figures 2b,d compare the performance of AuGMEnT, *re*-AuGMEnT, and the methods tested in [4]: RL-LSTM, Memory Bits [3] 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

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

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.

4 Discussion

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 [8]. This was resolved for LSTM by coupling memory elements to special forget-gates that can reset the memory and the associated gradients in [8]. 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 [3, 6], but neither of these models is biologically plausible. The PBWM model [5], 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 [10].

References

[1] Gottlieb and Goldberg. Activity of neurons in the lateral intraparietal area of the monkey during an antisaccade task. *Nat. Neurosci.*, 1999.

[2] Yang and Shadlen. Probabilistic reasoning by neurons. *Nature*, 2007.

[3] Peshkin, Meuleau, and Kaelbling. Learning Policies with External Memory. *ICML*, 1999.

[4] Bakker. Reinforcement learning with long short-term memory. In *NIPS*, 2002.

[5] O'Reilly and Frank. Making working memory work: a computational model of learning in the prefrontal cortex and basal ganglia. *Neural Comp.*, 2006.

[6] Todd, Niv, and Cohen. Learning to use working memory in partially observable environments through dopaminergic reinforcement. *NIPS*, 2009.

[7] Rombouts, Bohte, and Roelfsema. Neurally Plausible Reinforcement Learning of Working Memory Tasks. *NIPS*, 2012.

[8] Gers, Schmidhuber, and Cummins. Learning to forget: Continual prediction with LSTM. *Neural Comp.*, 2000.

[9] Sutton and Barto. *Reinforcement Learning: an introduction*. MIT Press, 1998.

[10] Fujii and Graybiel. Representation of action sequence boundaries by macaque prefrontal cortical neurons. *Science*, 2003.

[11] Wiering and Schmidhuber. HQ-learning. *Adaptive Behavior*, 1997.