ReScience

[Re] How Attention Can Create Synaptic Tags for the Learning of Working Memories in Sequential Tasks

Le Masson Erwan[2, 1] **and Alexandre Frédéric**[1, 2, 3]

1 LaBRI, Université de Bordeaux, Bordeaux INP, Talence, France **2** INRIA Bordeaux Sud-Ouest, Talence, France **3** Institut des Maladies Neurodégénératives, Université de Bordeaux, Bordeaux, France

frederic.alexandre@inria.fr

Editor
Name Surname

Reviewers
Name Surname
Name Surname

Received Sep, 1, 2015
Accepted Sep, 1, 2015
Published Sep, 1, 2015

Competing Interests:
The authors have declared that no competing interests exist.

Article repository

Code repository

A reference implementation of

→ How Attention Can Create Synaptic Tags for the Learning of Working Memories in Sequential Tasks, J. Rombouts, M. Bohte and P. Roelfsema, PLoS Computational Biology 11.3, e1004060. DOI: 10.1371/journal.pcbi.1004060

Introduction

The reference paper [2] introduces a new reinforcement learning model, called AuGMEnT, using memory units to learn sequential tasks. The results presented suggest new approaches in understanding the acquisition of tasks require working memory and attention, as well as biologically plausible learning mechanisms. The model improves on previous reinforcement learning schemes by allowing tasks to be expressed more naturally as a sequence of inputs and outputs. An implementation of the model was provided by the author which helped verifying the correctness of computations. The script written for this replication uses Python 3 along with NumPy and the multiprocessing libraries for speedup. Object oriented architecture is used in the module to help factorize the code.

Methods

Model

The initial intention was to implement the model using an artificial neural network simulator. The simulation tool ANNarchy [3] was considered for its ability to simulate rate-coded networks. Unfortunately, there were several incompatibilities with AuGMEnT, most notably the discrete time steps limiting its flexibility, and the order of evaluation in the simulator, i.e. connexions then populations, making it difficult to implement backpropagation models. The use of ANNarchy was abandoned and it was instead decided to write a custom script to simulate the network.

The paper's description of the model details all the update functions and is relatively straight forward to implement, only the initial value of $Q_a(t-1)$ was not provided for the first computation of δ in equation 17. Where the author's implementation uses $Q_a(t)$, it was decided to use a more naive solution and set it to 0. It might also be useful to clarify the nature of the feedback weights w' in equations 14 and 16: once an action is selected, only the feedback synapses leaving the corresponding selected Q-value unit are activated to update tags, more precisely: $w'_{ij} = w_{ij} \times z_i$. The model can have specific feedback synapses as well but the simpler method is to use the feedforward synapses' weights.

To offer some discussion about the model and its limits, the first point to bring forward would be its artificial time management. The extreme discretization of time and explicit signals such as trial begin and end make it difficult to consider real-time simulation or even realistic environments implementations. The difficulties of implementing AuGMEnT inside the simulator ANNarchy is a good illustration of those limitations. The authors are currently working on a continuous version of the model to address those issues. Another possible weakness worth noting is the ambiguity of some memory traces. Because the traces in memory units are the sums of changes in input, there exist sequences of inputs the model would be incapable of distinguishing. For example, the sequences `((0, 0), (1, 0), (1, 1))` and `((0, 0), (0, 1), (1, 1))` have the same memory traces `(1, 1)` and could lead to unintended results.

Tasks

The descriptions of the tasks used to test the network are somewhat minimal and it is sometimes necessary to read the refer to the original experiments' papers [1] and [4] for clarifications. In this section, some details of implementation will be exposed.

For the fixation tasks , i.e. saccade/anti-saccade and probabilistic decision making, once the network has fixated the point, it has to maintain fixating until the "go" signal where the screen turns off, otherwise the experiment is failed. Moreover, during the "go" phase, the gaze can only be chosen once, if it is not the target, the trial fails. At the end of the trial, a blank screen is shown before exiting.

The shaping strategy for the probabilistic decision making task consists in increasing gradually the difficulty of the task. The table 3 in the article describes all 8 levels of difficulty. The column "# Input Symbols" is the size of the subset of shapes. The network is not confronted to all shapes immediately: first, the two infinite weight shapes are used, then shapes with the smallest absolute weights are added as the difficulty increases. The column "Sequence Length" is the number of shapes shown during a trial. The more shapes there are on screen, the more difficult it is to determine which target should be chosen. A number of settings are randomized, such as which shapes should appear, their order of apparition, but also their locations around the fixation point. The first shape can appear in any of the 4 locations, the second in any of the remaining 3 locations, etc. Finally, the triangle and heptagon, shapes with infinite weights, cancel each other in the computation of the total weight.

Results

Only the saccade/anti-saccade task and the probabilistic decision making task were implemented. The results observed, presented in table 1, are comparable to the original article's. The differences could come from undocumented changes in the experiments' protocols. Qualitative results, for example the use of shaping strategy to obtain better performances, are confirmed by this replication.

Conclusion

The results obtained are comparable to those announced in the article. Ambiguities in the experiments' descriptions could be the cause for worse performances, but do not contradict the article's overall conclusion.

Table 1: Results

Task	Success in [2]	Success	Convergence in [2]	Convergence
Saccade w/ shaping	99.45%	85.0%	4100 trials	3800 trials
Saccade w/o shaping	76.41%	64.0%	N/A	5433 trials
Probabilistic	99.0%	100%	55234 trials	70762.5 trials

References

[1] J. Gottlieb and M. Goldberg. "Activity of neurons in the lateral intraparietal area of the monkey during an antisaccade task". In: *Nature Neuroscience* 2.10 (1999), pp. 906–12. DOI: 10.1038/13209.

[2] J. Rombouts, S. Bohte, and P. Roelfsema. "How Attention Can Create Synaptic Tags for the Learning of Working Memories in Sequential Tasks". In: *PLoS Computational Biology* 11.3 (2015), e1004060. DOI: 10.1371/journal.pcbi.1004060.

[3] J. Vitay, H. Dinklebach, and F. Hamker. "ANNarchy: a code generation approach to neural simulations on parallel hardware". In: *Frontiers Neuroinformatics* 9.19 (2015). DOI: 10.3389/fninf.2015.00019.

[4] T. Yang and M. Shalden. "Probabilistic reasoning by neurons". In: *Nature* 447.7148 (2007), pp. 1075–80. DOI: 10.1038/nature05852.