Project Paper

Andrew Hampton

Biologically plausible learning in recurrent neural networks reproduces neural dynamics observed during cognitive tasks

Thomas Miconi
eLife 6 (2017): e20899
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5398889

Summary: In this paper, the author presents a novel algorithm for training a recurrent neural network. The algorithm seeks to more accurately represent biological systems than those typically used in machine learning applications. Artificial neural networks have the unrealistic advantage (from a biological perspective) of receiving constant, real-time supervised rewards. By contrast, the author asserts that a biologically plausible network must learn from temporally sparse, discrete rewards.

I will only summarize the author's learning rule and the results of my implementation of one particular task. However, in the paper, the author offers a more dramatic claim: that algorithms of the sort presented here offer a model consistent with the learning observed in animal behavior.

Learning Rule: The algorithm is based on a recurrent neural network given by the differential equation:

$$\tau \frac{dx_i}{dt} = -x_i(t) + \sum_{j=1}^{N} J_{ij} r_j(t) + \sum_{k=1}^{M} B_{ik} u_k(t).$$

Here, $\mathbf{x}$ represents the excitation of the neurons and $\mathbf{r}$ the responses. J is an adjacency matrix giving the weights of the internal network, while B represents the weights associated with the external inputs $\mathbf{u}$.

The response function used in all of the experiments is:

$$\mathbf{r}(t) = \tanh(\mathbf{x}(t)).$$

The particular values of the constants can be found in the paper or the author's code [1]. Most noteworthy, perhaps, is the initialization of the weight matrix J. The entries are initially taken from a normal distribution with mean zero and standard deviation $\frac{g}{\sqrt{N}}$, where N is the number of neurons in the network and g is a constant governing the *chaotic* behavior of the network. The value of g used in these experiments is 1.5, which allows the network to exhibit chaotic behavior.

At every time step of a trial the weight change accumulates in what the author calls a *potential* weight change, given by:

$$e_{ij}(t) = e_{ij}(t-1) + S(r_j(t-1) \cdot (x_i(t) - \overline{x}_i)).$$

Here, S is a monotonic, supralinear function (quoting the author: *the plasticity mechanism is dominated by large increments, and tends to suppress smaller ones*) and $\overline{\mathbf{x}}$ is a short-term running average of $\mathbf{x}$.

The actual weight update is given by either

$$\Delta J = \eta e (R - \overline{R})$$

or

$$\Delta J = \eta e (R - \overline{R}) \overline{R}.$$

In these equations, η represents the learning rate, R the trial reward, and $\overline{R}$ a running average of the rewards for a given trial type. See the variations on the task below for a more detailed discussion of the implementation of this $\overline{R}$.

The first version above appears in the paper while the latter is used in the author's code. Below, we show the results of solving a given task with both versions.

Delayed XOR Task: The author presents several tasks, the simplest of which is a basic classification task. Two binary stimuli, whose values are referred to as A and B, are presented sequentially, and a designated output neuron is trained to emit the value 1 if the stimulus sequence is AA or BB and to emit -1 in the other two cases.

The trial reward (i.e., penalty) is calculated by taking the mean absolute deviation from the expected output over the last 200 timesteps (where each timestep simulates one millisecond).

I replicated the results of this task and performed a few experiments with variations on the parameters [2]. In Figure 1, the results of four variations are presented:

1. The same task as described in the paper. A table of the reward running average is kept with a separate running average for each of the four trial types.

2. A modified version of the learning rule (the second learning rule given above). Note that the convergence is much more stable in this case.

3. In this version, only two running averages of the reward are kept: one for each of the output cases. We don't observe any learning in this case.

4. Finally, a version which uses a discretized version of the reward: no penalty if the output is on average close to the desired output over the final 200 timesteps, otherwise a unit penalty. The discrete reward was ineffective.

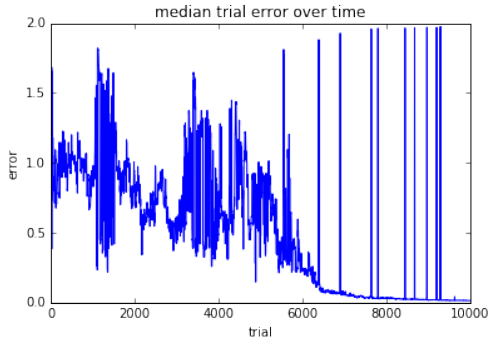
In Figure 2, we see the response of the output neuron over time, taken from the second experimental case described above. The sequence of stimuli after training was AB, and we observe the correct behavior.

Finally, Figures 3 and 4 show the results of a modification to this task. Keeping a table of running averages of the rewards is somewhat artificial. Instead, we can train four neurons to output the characteristic function of the trial type. Then, we can update all of the running averages each trial, weighted by the respective output neurons.

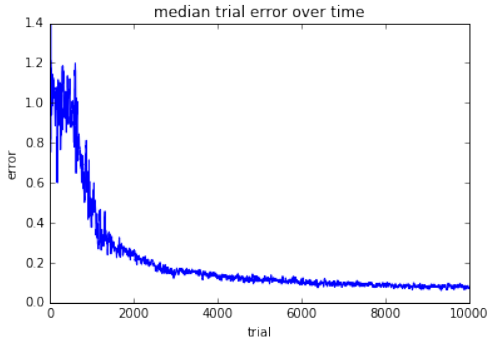
For this alternate formulation of the task, the error was computed by determining which of the four output neurons on average responded closest to the value 1 over the last 200 timesteps, then taking the mean absolute deviation of that neuron from 1. We see that this was model is also successful in learning the task.

Links:

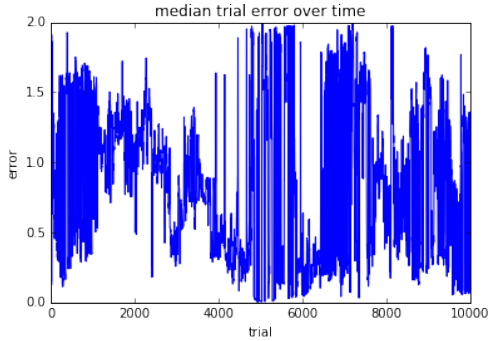
[1] https://www.github.com/ThomasMiconi/BiologicallyPlausibleLearningRNN
[2] https://www.github.com/awhampton/math510



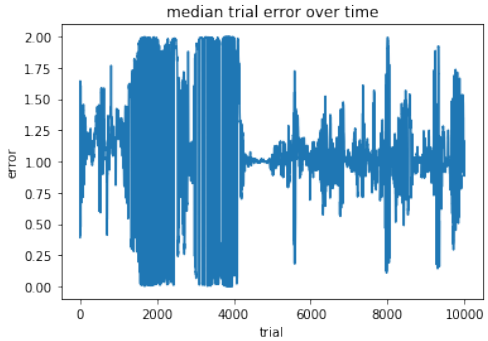
(a) $\Delta J = \eta e(R - \overline{R})$



(b) $\Delta J = \eta e(R - \overline{R})\overline{R}$

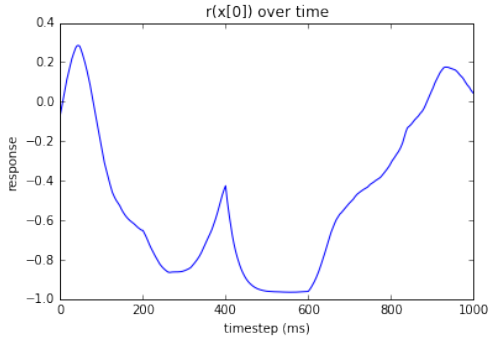


(c) Binned learning rule

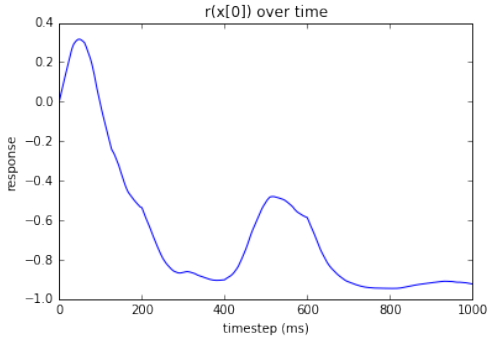


(d) Discrete learning rule

Figure 1: Error, taken as a rolling median over twenty trials.



(a) Before training



(b) After training

Figure 2: Output neuron before and after training during the task of Figure 1b. Observe that, after training, the network has learned to distinguish trial type AB.

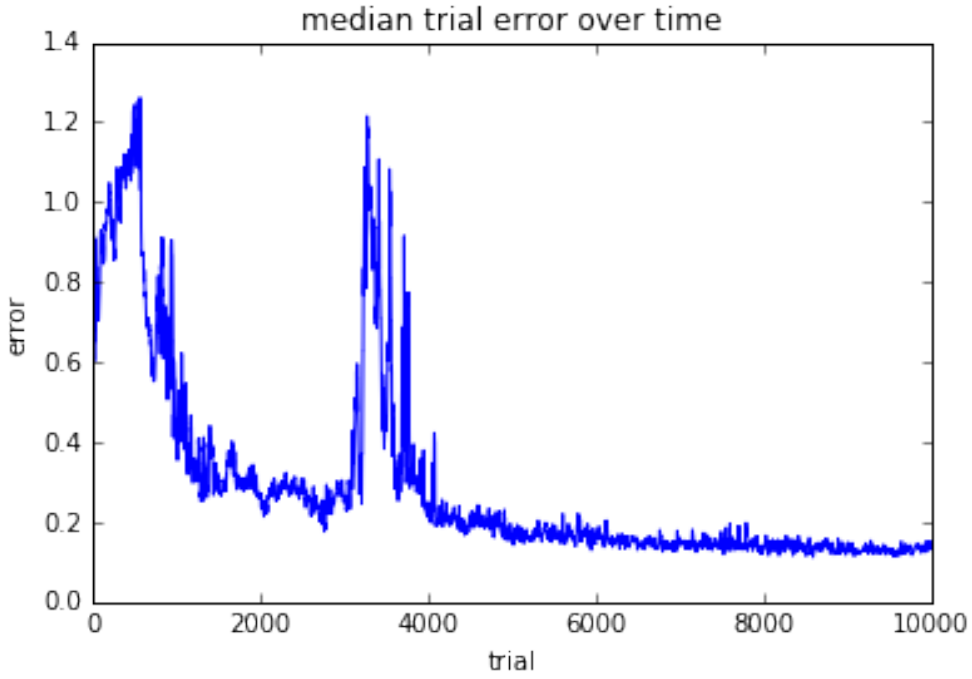
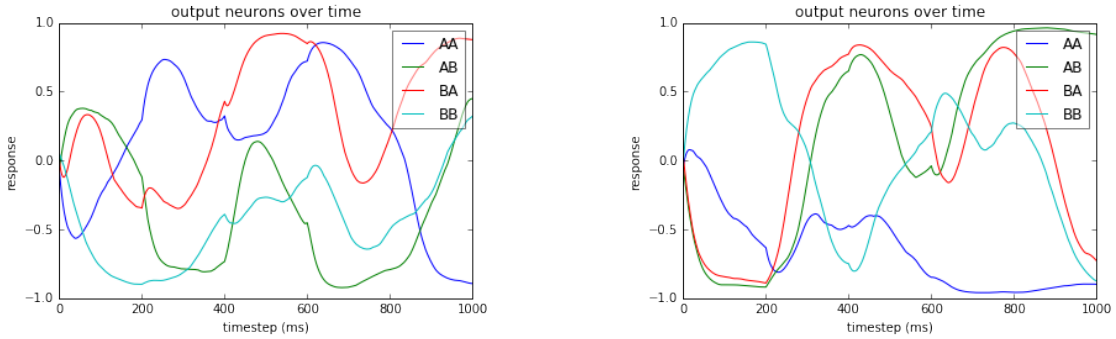


Figure 3: Error in alternate formulation of the task, where the characteristic function of the trial type is learned.



(a) Before training

(b) After training

Figure 4: Output neurons before and after training during the task of Figure 3. Observe that, after training, the network has learned to distinguish trial type AB.