Incorporating an adaptive learning rate in a neural model of action selection

Sverrir Thorgeirsson (sverrir.thorgeirsson@uwaterloo.ca)
Brent Komer (bjkomer@uwaterloo.ca)
Chris Eliasmith (celiasmith@uwaterloo.ca)
Centre for Theoretical Neuroscience, University of Waterloo 200 University Avenue West,
Waterloo, ON, Canada, N2L 3G1

Abstract

In previous work, we have implemented spiking neuron models that use a biologically realistic action selection system to solve complex cognitive tasks, including the Tower of Hanoi and semantic memory search. However, such models often require the fine-tuning of multiple parameters so that the model can reach a desired level of performance. Recently, we demonstrated that a local, online learning rule, which only requires a single parameter, the learning rate, is sufficient for teaching our model how to solve a general cognitive sequencing task. Here, we refine our method by showing that adding adaptive learning is more robust regarding our choice of parameters, and will achieve better performance on all versions of the cognitive task that we tested. These results provide a foundation for building complex cognitive models that require no hand-tuning of parameters.

Keywords: neural engineering framework; neural production systems; spiking neurons; metaplasticity; adaptive learning rate; basal ganglia; semantic pointer architecture

Introduction

The work in our research group is centred on using spiking neuron models to build biologically realistic models that can perform complex cognitive tasks. Our group has constructed Spaun (Eliasmith et al., 2012), a very large-scale (2.2 million neuron) model that can perform multiple cognitive tasks, but we have also built models that can solve the Tower of Hanoi, perform sentence parsing, solve bandit tasks and so forth, all using the Neural Engineering Framework (NEF; Eliasmith & Anderson, 2003) and Semantic Pointer Architecture (SPA; Eliasmith, 2013).

Until recently, our approach to building spiking neural models that can solve cognitive tasks has required us to hand-tune of a set of parameters before the models can be initialized. This can be both challenging and time-consuming, which is why we showed in a soon-to-be-published paper (Stewart, Thorgeirsson, & Eliasmith, in press) how a local, online learning rule can be used to learn the required parameters. In a more recent paper (Thorgeirsson, Stewart, & Eliasmith, in press), we showed that this approach can also be applied on larger version of the original cognitive task that we tested, but it still requires us to specify a one-dimensional parameter called the learning rate.

Our results showed that although the approach is easier than tuning a large number of parameters, finding a learning rate can be difficult. First, there is a narrow range of learning rates such that the model will perform well. Second, the learning rate is not robust regarding the size of the task; certain values will perform well for large versions of the task and poor for small versions, and vice versa.

Our goal with this paper is to present an updated version of the algorithm using an adaptive learning rate, in order to solve the two aforementioned problems. Furthermore, we wish to use a biologically implementable version of the learning rule algorithm, considering that there is evidence from neuroscience that the brain contains mechanisms for modulating its rate of learning while an animal is performing a task (Farashahi et al., 2017; Abraham, 2008; Müller-Dahlhaus & Ziemann, 2015).

Neural Engineering Framework and Semantic Pointer Architecture

The NEF provides a means of producing a spiking neural network from a mathematical description of a desired function. This allows the creation of biologically plausible models based on the theory of the functions being computed in the brain.

The core principles of the NEF are representation, transformation, and dynamics (Eliasmith & Anderson, 2003). The first principle provides a means of using neural activity as a basis for representing vectors of arbitrary dimensionality. Representation is achieved through nonlinear encoding via the tuning curves of the neurons and a weighted linear decoding. The second principle allows transformations of the represented information through an alternate weighted linear decoding that approximates the desired function. The third principle states that neural representations can be analyzed as control theoretic state variables and control theory can be applied to this time varying dynamic system.

The SPA specifies several architectural components and representational strategies for building spike-based cognitive models. Of particular interest to this work is the basal ganglia-thalamus-cortex loop, which we focus on as the main sequencing controller, and whose use is described more in the next section. In addition, we use the symbol-like representations, semantic pointers, suggested by the SPA to represent cognitive states, inputs, and outputs. We identify these representations with all caps (e.g., 'ONE'), which should be interpreted as the name of a high-dimensional vector that is represented in spiking neurons. All processing in the model is performed at the level of spiking neurons.

Neural Action Selection using a Basal-Ganglia Model

To perform action sequencing, we use a neural implementation of the cortex-basal ganglia-thalamus system (Stewart, Choo, & Eliasmith, 2010). The model consists of visual information, a working memory, the state of the motor system, and other critical components of the human neural action selection system. To compute the utility of actions, we use the connections between the cortex and the striatum in the basal ganglia. The basal ganglia will then select the largest of those utility values, and the corresponding action will be selected via connections from the basal ganglia to the thalamus.

Cognitive Task

In a previous paper (Stewart et al., in press), we introduced a simple cognitive task for testing a learning approach. The cognitive task has two cortical state vectors called LETTER and NUMBER and six actions called A, B, C, ONE, TWO and THREE. When the system is run, the model's visual system (v) will alternate every second between the symbols LETTER and NUMBER via an external input. Meanwhile, the model's working memory state (m) should cycle through the symbols A, B and C or ONE, TWO and THREE, respectively.

The expected behavior can be modeled by the following production system:

IF	**THEN**
$v =$ LETTER AND $m =$A	$m =$ B
$v =$ LETTER AND $m =$B	$m =$ C
$v =$ LETTER AND $m =$C	$m =$ A
$v =$ LETTER AND $m =$ONE	$m =$ TWO
$v =$ LETTER AND $m =$TWO	$m =$ THREE
$v =$ LETTER AND $m =$THREE	$m =$ ONE

To exhibit this behavior in a neural model, we organize the neurons to compute six utility functions (one for each action), and then select the action with the highest utility. The following six equations can be used:

$$U_1 = \alpha_1 \cdot v \cdot \text{LETTER} + \beta_1 \cdot m \cdot A$$
$$U_2 = \alpha_2 \cdot v \cdot \text{LETTER} + \beta_2 \cdot m \cdot B$$
$$\dots$$
$$U_6 = \alpha_6 \cdot v \cdot \text{NUMBER} + \beta_6 \cdot m \cdot C$$

In order to find the optimal parameters α_i and β_i of the utility functions without hand-tuning them before the model is run, we proposed that an online learning rule could be used. The rule is based on the delta rule (Widrow & Hoff, 1960), which can be described in this way:

$$\Delta\omega_{i,j} = \alpha x_i(t_j - y_j)$$

where $\omega_{i,j}$ are weights on values x_i that produce y_j. The target value is t_j and the learning rate is α. The parameters are initialized to zero. This implies that then adjusted based on the model's performance, using a supervised signal.

There exist two options for calculating the output y_j; first, it can equal the represented output of the neural system (i.e. $y_j = \sum_i \omega_{i,j} x_i$), which is in the range $[0,1]^6$. The second option is to choose the output of the basal-ganglia action selection system (i.e., $y_j = 1$ if the action j is selected, otherwise $y_j = 0$).

For the learning input x_i, there are also two options. The first option is that we use the base terms from the aforementioned equations that determine the utility functions U_i. Alternatively, we can let x_i represent the neural activity in the cortical neurons that are calculating the utility functions. The second approach often gives better performance since it increases the range of possible functions that can be found by our learning rule, however, it means that the number of parameters increases substantially, so for this paper we chose the former approach.

In a second paper (Thorgeirsson et al., in press), we expanded this task to test the scalability of the learning rule in the context of our cognitive model. To that end, we let the number of cortical state variables shown in the visual system, called alphabets, equal some positive integer n, and the number of symbols in each alphabet equal a positive integer m, so that the number of actions equals $n \cdot m$. We call the resulting model a generalized sequencing task (n,m) for short. For example, the generalized sequencing task $(3,4)$ may have the state variables LETTER, NUMBER and ORDINAL and the actions A, B, C, D, ONE, TWO, THREE, FOUR, FIRST, SECOND, THIRD, FOURTH.

Gradient descent and adaptive learning rate algorithms

The results of applying the spiking delta rule on the generalized sequencing task suggest that this learning rule is not robust regarding the learning parameter α. In a version of the task that contains five alphabets with five symbols each (i.e., 25 actions in total), the model did not learn the task successfully when the learning rate was set to 10^{-9} and 10^{-11}, but it managed to do so in a reasonable time frame for the learning rate 10^{-10}. Furthermore, our results showed that different versions of the task required different learning rates; when the model contains ten actions, our results showed that the model is on average eight times as fast in solving the task for the learning rate 10^{-9} than for the learning rate 10^{-10}, even though using the former learning rate did not perform well for a higher number of actions.

We would like to use a biologically plausible variant of our delta learning rule that also adjusts the parameter α based on the model performance. That way, we hope to be able to successfully and quickly perform different versions of the generalized sequencing task that contains an arbitrary amount of actions. Furthermore, the learning rule should be robust regarding the choice of any hyperparameters.

In recent years, several adaptive learning rate algorithms have been proposed in machine learning literature and have been used successfully on real-life problems. Four of those include ADAGRAD (Duchi, Hazan, & Singer, 2011) an algorithm which divides the learning rate α by the root of the sum of squares (RSS) of previous gradients; RMSProp (Tieleman & Hinton, 2012), which we used in this work and is defined below; ADADELTA (Zeiler, 2012), which is similar to RMSProp but does not require a learning rate; and ADAM (Kingma & Ba, 2014), which is an improvement over RMSProp according to its authors, but in our view, it is harder to implement in a biologically realistic setting since it requires the computation of both first and second moment estimates.

The RMSProp algorithm divides the learning parameter α by the root of the exponentially decaying average of squared gradients that have been calculated at the previous timesteps plus a small number to avoid division by zero. In other words, we divide α by the value

$$\sqrt{\sum_{i=1}^{t} g_i^2 \cdot a^{t+1-i} + \varepsilon}$$

where g_i is the gradient at time step i for a single parameter. According to Tieleman and Hinton, the values a and ε can be set to the default values of 0.9 and 10^{-3} respectively, which is what we did in this paper.

Experimental Setup

When simulating the generalized sequencing task (n,m) under our implementation of RMSProp, we follow the same protocol as in our previously documented experiments; we let each cycle consist of a presentation of all m alphabets in the visual system and alternate between cycles where learning is on and where learning is off (i.e., on-off learning).

To evaluate the model performance, we use the same metric as in our most recent paper; we return the index of the first cycle where the model manages to consecutively complete every transition in every alphabet. For instance, for the model $(2,2)$ with the symbols A, B and ONE, TWO, we would return the index of the first cycle in which the model manages to successfully complete the round $A \rightarrow B \rightarrow A$ when the visual system is showing LETTER *and* completes ONE $\rightarrow$ TWO $\rightarrow$ ONE when the visual system is showing NUMBER.

We collected the average result of running multiple iterations of the results of the model for different random seeds, which affect how our neural simulator sets various properties of the neurons such as maximum firing rates and tuning curves. We let the model perform the tasks (n,m) for $n,m \in [3,4,5]$, which gives nine differently sized versions of the task and means that the number of actions in each task is in the range $[9,16]$.

We have previously observed that once this evaluation metric indicates that the task has been learned, the model will continue to perform correctly in all subsequent trials. This justifies our choice of metric.

Results

The results of our experiment can be seen in Figure 1. For our baseline delta rule algorithm, we found that when setting the learning rate α to 10^{-2} and 10^{-4}, and for any values that are higher or lower than those, respectively, the model performed poorly or was unable to learn any version of the task in many cases. However, for $\alpha = 10^{-3}$, the model was able to learn most of the tasks in each simulation trial, as can been seen in the figure.

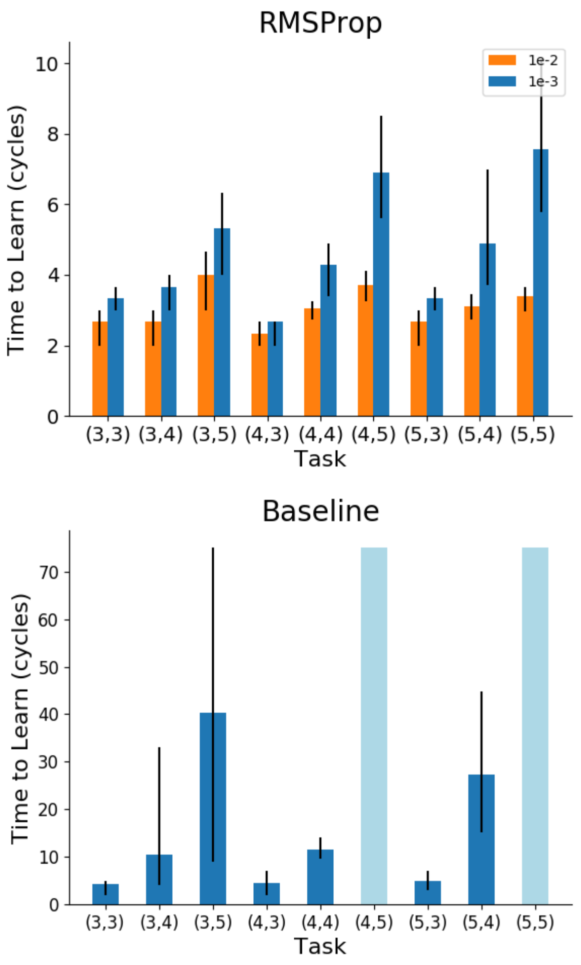

Figure 1: The top figure shows how the RMSProp algorithm performed with 95% bootstrap intervals for $\alpha = 10^{-2}$ and $\alpha = 10^{-3}$ using our evaluation metric. The bottom figure shows the performance of our baseline spiking delta rule algorithm under the $\alpha = 10^{-3}$. For the baseline version, the model did not learn the task within our stopping point 75 cycles for the tasks $(4,5)$ and $(5,5)$ in the majority of the simulations, so the mean value of the evaluation results could not be computed.

For the RMSProp algorithm, setting α to 10^{-2} and 10^{-3} gave better results on average than the baseline algorithm did for any value of α and for any version of the task, with the exception of the easier tasks $(3,3)$ and $(4,3)$ where there was no statistical performance difference. For RMSProp, we also tried the parameters $\alpha = 10^{-1}, 10^{-4}$, which gave significantly worse results but were nevertheless sufficient for learning each version of the task.

Interestingly, the results also show that for RMSProp with the optimal value of α (10^{-2}), the model will solve tasks of any size in the narrow range of two to four cycles on average. In other words, the size of the task does not appear to be a large performance factor here in contrast with the baseline algorithm, where the task size will impact not only how fast the model will learn the task but also whether the model can solve it at all.

Discussion and Future Work

The results show that by using the adaptive learning rate algorithm RMSProp, we can solve both of the problems that we had previously experienced.

First, we found that there is a fairly wide range of RMSProp parameters that not only gives adequate results for all versions of the generalized sequencing tasks that we implemented, but also performs much better than the baseline algorithm with our version of the delta rule. This indicates that in the future, we may be able to avoid manual tuning of model parameters.

Second, we found that this algorithm is robust when it comes to different versions of the sequencing tasks; we found no RMSProp parameters that give particularly good results for small versions of the task and poor results for large versions of the task, or vice versa. This indicates that for cognitive models that we may implement in the future, we may be able to perform generalizations of the tasks without having to adjust any parameters.

Using this new updated learning rule, we were able to solve tasks that our model had previously struggled with with a very large performance improvement. It would be interesting to see whether this performance holds up for cognitive tasks where the number of actions is very high (> 100) and whether there exist some generalized sequencing task (n,m) that the model is still not able to solve.

Aside from that, our work can be extended in two directions. First, we would like to apply the same algorithm on more complex cognitive tasks such as our existing Tower of Hanoi model (Stewart & Eliasmith, 2011), but also other large-scale cognitive tasks that we have not yet implemented. Second, we intend to see if we can adjust the learning scheme used in this model so that we no longer need a supervised signal, but can instead use reinforcement learning with the evaluation metric as a reward function. We believe that the results documented in this paper provides a foundation for both approaches.

Acknowledgments

The authors would like to thank Terrence Stewart for discussions that helped improve this paper. This work was supported by CFI and OIT infrastructure funding as well as the Canada Research Chairs program, NSERC Discovery grant 261453, AFOSR grant FA9550-17-1-0026 and NSERC graduate funding.

References

Abraham, W. C. (2008). Metaplasticity: tuning synapses and networks for plasticity. *Nature Reviews Neuroscience*, *9*(5), 387.

Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, *12*(2), 2121–2159.

Eliasmith, C. (2013). *How to build a brain* [book]. Oxford: Oxford University Press.

Eliasmith, C., & Anderson, C. H. (2003). *Neural engineering: Computation, representation, and dynamics in neurobiological systems* [book]. Cambridge, MA: MIT Press.

Eliasmith, C., Stewart, T., Choo, X., Bekolay, T., DeWolf, T., Tang, Y., & Rasmussen, D. (2012). A large-scale model of the functioning brain. *Science*, *388*, 1202-1205.

Farashahi, S., Donahue, C. H., Khorsand, P., Seo, H., Lee, D., & Soltani, A. (2017). Metaplasticity as a neural substrate for adaptive learning and choice under uncertainty. *Neuron*, *94*(2), 401–414.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *CoRR*, *abs/1412.6980*. Retrieved from `http://arxiv.org/abs/1412.6980`

Müller-Dahlhaus, F., & Ziemann, U. (2015). Metaplasticity in human cortex. *The Neuroscientist*, *21*(2), 185–202.

Stewart, T. C., Choo, X., & Eliasmith, C. (2010). Dynamic behaviour of a spiking model of action selection in the basal ganglia. In *10th int. conf. on cognitive modeling.*

Stewart, T. C., & Eliasmith, C. (2011). Neural cognitive nodelling: A biologically constrained spiking neural model of the Tower of Hanoi task. In *33rd annual meeting of the cognitive science society.*

Stewart, T. C., Thorgeirsson, S., & Eliasmith, C. (in press). Supervised learning of action selection in cognitive spiking neuron models. In *Proceedings of the 40th annual conference of the cognitive science society.*

Thorgeirsson, S., Stewart, T. C., & Eliasmith, C. (in press). Analysis of learning action selection parameters in a neural cognitive model. In *International conference on cognitive modelling 2018.*

Tieleman, T., & Hinton, G. (2012). *Lecture 6.5 - RMSPROP* (Tech. Rep.). Coursera.

Widrow, B., & Hoff, M. E. J. (1960). Adaptive switching circuits. *IRE Western Electric Show and Convention Record, Part 4*(Part 4), 96–104.

Zeiler, M. D. (2012). Adadelta: An adaptive learning rate method. *CoRR*, *abs/1212.5701*. Retrieved from `http://dblp.uni-trier.de/db/journals/corr/corr1212.html\#abs-1212-5701`