Parameter exploration of a neural model of state transition probabilities in model-based reinforcement learning

Mariah Martin Shein (mshein@uwaterloo.ca)
Terrence C. Stewart (tcstewar@uwaterloo.ca)
Chris Eliasmith (celiasmith@uwaterloo.ca)
Centre for Theoretical Neuroscience, University of Waterloo
Waterloo, ON, Canada, N2L 3G1

Abstract

We explore the effects of parameters in our novel model of model-based reinforcement learning. In this model, spiking neurons are used to represent state-action pairs, learn state transition probabilities, and compute the resulting Q-values needed for action selection. All other aspects of model-based reinforcement learning are computed normally, without neurons. We test our model on a two-stage decision task, and compare its behaviour to ideal model-based behaviour. While some of these parameters have expected effects, such as increasing the learning rate and the number of neurons, we find that the model is surprisingly sensitive to variations in the distribution of neural tuning curves and the length of the time interval between state transitions.

Keywords: neural model; reinforcement learning; model-based reinforcement learning; Neural Engineering Framework

Introduction

Reinforcement learning (RL), a formalization of reward-based decision making, is often divided into two sub-types: model-free and model-based (Sutton & Barto, 1998). This distinction has been used by neuroscientists to explain aspects of instrumental conditioning in humans and other animals. Daw, Niv, and Dayan (2005) drew parallels between the habit system (where actions are performed automatically) and model-free RL; and between the goal-directed system (where actions show evidence of planning) and model-based RL. Model-free and model-based learning have been proposed to be realized in the brain with separate systems that rely on different prediction error signals (Glascher, Daw, Dayan, & O'Doherty, 2010). There has been extensive research on model-free RL, including work on how it may be instantiated in the brain according to the reward prediction error theory of dopamine (e.g., Barto, 1995). There is significantly less focus on model-based RL, including a particularly evident lack of suggestions as to how it may happen in the brain (Friedrich & Lengyel, 2016).

We have developed a novel model of model-based RL that uses spiking neurons to represent state-action pairs, learn state transition probabilities, and compute the resulting Q-values needed for action selection. The present work aims to investigate the factors that influence the behaviour of this neural model.

Background

In both model-free and model-based RL approaches, the goal is to learn from experience how valuable different actions are, given the current state (Sutton & Barto, 1998). This is written as $Q(s,a)$, where s is the state, and a is the action.

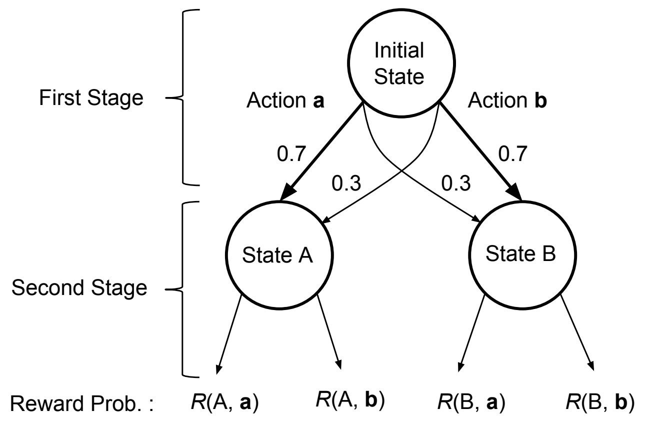


Figure 1: Schematic diagram of the two-stage task. The numbers on the arrows in the first stage indicate the probability of a particular state transition, given the chosen action. The reward after performing an action in the second stage is randomly determined based on the reward probability. Adapted from Akam et al. (2015).

In model-based RL, the value Q of an action in a particular state is given by the Bellman equation (Bellman, 1957):

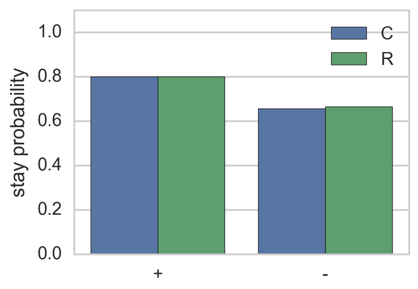
$$Q(s,a) = R(s,a) + \gamma \sum_{s'} P(s,a,s') \max_{a} Q(s',a), \tag{1}$$

where R is the expectation of reward, P is the probability of transitioning from state s to s' if action a is performed, and γ is a future discounting parameter. Importantly, in order to compute this, the system needs to know the state transition probabilities $P(s,a,s')$. This set of probabilities can be thought of as a model of the environment, and is why this approach is called model-based.

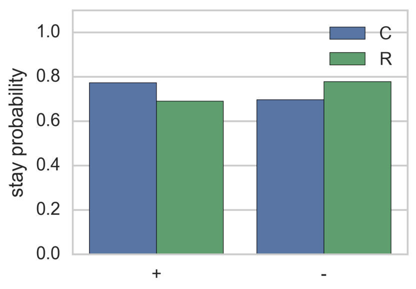
In contrast, model-free RL does not create an explicit representation of the environment. Rather, it just uses whatever state s_{t+1} occurs after the action a in the current state s_t. This can be thought of as an estimate of the typical next state that will occur. This leads to an approximation of Eq. 1 that does not take into account the transition probabilities, but is much simpler to compute:

$$Q(s_t,a) = R(s_t,a) + \gamma \max_{a} Q(s_{t+1},a). \tag{2}$$

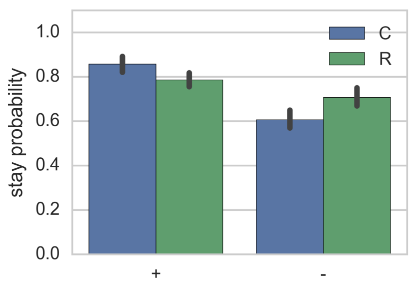
Model-free RL constructs an estimate of Eq. 1 through direct experience in the environment, which leads to an implicit



(a) Ideal model-free behaviour



(b) Ideal model-based behaviour



(c) Human behaviour

Figure 2: Ideal and human stay probability behaviour on the two-stage task. C denotes common, and R denotes rare, state transitions. Trials are either rewarded (+) or unrewarded ($-$). Adapted from Daw et al. (2011).

representation of environmental statistics. In contrast, model-based RL constructs an internal model of the probabilities of reward R and state transitions P. This explicit learning of the statistics is used to directly calculate the Bellman equation.

Given this difference, there are situations where the learning trajectories will differ between model-free and model-based RL. As discussed in the next section, particular learning tasks can be defined to distinguish the two approaches.

Two-stage task

We test our model on the two-stage task described in Daw et al. (2011). A schematic diagram of this task is shown in Fig. 1. The first stage consists of an initial state, which has two possible actions (**a** and **b**). These actions lead probabilistically to one of the two second stage, or terminal, states (A and B), with action **a** commonly transitioning to state A and action **b** commonly transitioning to state B. These common state transitions each have a probability of 0.7; correspondingly, rare transitions have a probability of 0.3. In states A and B, actions **a** and **b** are again available, and are rewarded with probability determined by a Gaussian random walk, so that the immediate reward after performing an action in a second stage state is either 0 or 1. This randomness was introduced to enforce continued learning throughout the task.

This task was developed to discriminate model-based from model-free behaviours, using the stay probability, i.e., the likelihood of choosing the same initial-state action in trial $n+1$ as in trial n (Daw et al., 2011). In particular, if an agent finds itself in a rare (R) second-stage state (given their action in the initial state), and performs an action that is rewarded (+), a purely model-free strategy would increase the value of performing that first-stage action, as shown in Fig. 2a, while a purely model-based strategy would increase the value of being in that second-stage state, thus increasing the value of the unchosen first-stage action and decreasing the stay probability (see Fig. 2b, R+). By similar logic, in a rare, unrewarded state (R$-$), a model-based agent would increase the value of choosing the initial action, thus increasing the stay probability.

As an example of model-based reasoning, say an agent has found itself in state B after performing action **a**. The agent "knows" this is a rare transition. Now say the agent performs some action and receives a reward. This increases the value of being in state B, so the agent wants to return to this state. Since the agent knows that state B is more commonly reached after performing action **b** in the initial state, it will also increase the value of performing action **b** in the initial state and correspondingly *decrease* the value of performing action **a** in that state. Conversely, a model-free reasoner would simply increase the value of every state-action pair it performed before receiving the reward, and so the value of performing action **a** in the initial state would *increase*.

Daw et al. (2011) found that human behaviour on this task showed characteristics of both model-free and model-based strategies (see Fig. 2c). In particular, there is a statistically significant difference between rare-rewarded (R+) and common-unrewarded (C$-$) probabilities not evident in model-based behavior, and there are elevated rare-unrewarded (R$-$) and common-rewarded (C+) probabilities relative to the C$-$ probabilities that is not evident in model-free behaviour.

Model

Fig. 3 shows a schematic diagram of the model we built using the Nengo neural simulator (Bekolay et al., 2014), which is based on the principles of the Neural Engineering Framework (NEF; Eliasmith & Anderson, 2003). The NEF provides methods to generate neurons with random properties (such as tuning curve distributions) and then arrange them so that they best approximate a given representation or transformation.

The neural model includes two components of model-based RL: 1) the representation of the state transition probabilities $P(s,a,s')$, and 2) the multiplication of these probabilities by the Q-values of the future states to produce an estimate of the Q-values of the current state's actions. We implement these components using spiking leaky-integrate-and-fire (LIF) neurons (Lapicque, 1907) via the NEF, while the rest of the model-based RL system is implemented using traditional computation.

In the model, the states and actions are represented by vectors. For simplicity of explanation, we refer to these vectors as orthogonal, with the smallest possible dimensionality (three dimensions to represent the three states, and two for the two actions); however, our methods allow these vectors to be arbitrarily large. In the model, we use 5-dimensional vectors. For purposes of explanation, we will use 3-dimensional vectors, and assume state A is represented by $S_A = [1,0,0]$, state B by $S_B = [0,1,0]$, and the initial state by $S_0 = [0,0,1]$. The rest of the model components, shown in rectangles in Fig. 3, are implemented directly without neuron approximation. These components perform action selection, track and update the environment's actual state transitions and reward, learn the model-free Q-values of actions in states A and B, and store all the Q-values (for use in action selection).

In the design of our model, we exploit the natural parallelism of a neural implementation. In traditional model-based approaches, the state transition probabilities $P(s,a,s')$ are stored in lookup tables. However, in our model, these probabilities are represented by a function computed in a connection between neural populations that maps state-action inputs to a second-state probability distribution as follows:

$$P(s,a) = [P(s,a,S_A), P(s,a,S_B), P(s,a,S_0)]. \quad (3)$$

For example, $P(S_0,\mathbf{a}) = [0.7,0.3,0.0]$ in the two-stage task.

In a traditional model-based agent, the value of actions in states in the first stage (in this case, the single initial state) are recalculated at the beginning of each trial, with:

$$Q(s,a) = \sum_{s'} P(s,a,s') \max_a Q(s',a). \quad (4)$$

In our model, this calculation is done in neurons. Specifically, using our modified representation of the state transition probabilities, we calculate the following dot product:

$$Q(s,a) = P(s,a) \cdot Q(a'), \quad (5)$$

where $Q(a')$ is the vector of the best possible action for every state.

Although multiplications are non-linear, they have a well-characterized implementation in neurons that can be implemented accurately with the NEF (Gosmann, 2015). A neural population, called *Product* in Fig. 3, performs an element-wise multiply based on this characterization, and a summation is performed by the output connections to compute Eq. 5.

To illustrate how this would be done by a model-based agent using the two-stage task, consider an agent that is in the initial state and considering performing action **a** (i.e., $s = S_0, a = \mathbf{a}$). The agent remembers the Q-values of the best possible action a' for every state, say $Q(a') = [0.25,0.75,0.33]$, as well as the probability of reaching that state given the current state and action (as before, $P(S_0,\mathbf{a}) = [0.7,0.3,0.0]$). To calculate the Q-value of the current state and considered action, it performs the dot product of these two vectors, producing a value of $Q(S_0,\mathbf{a}) = 0.4$. It then follows the same process to consider action **b**, and finds $Q(S_0,\mathbf{b}) = 0.6$.

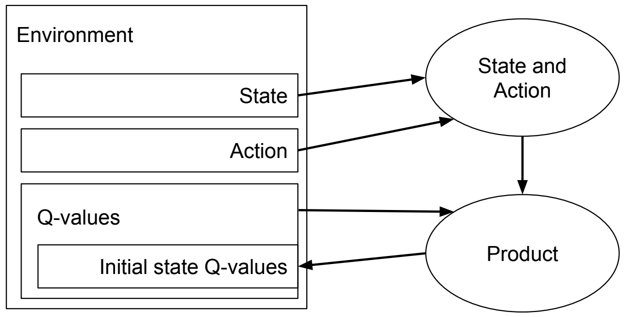


Figure 3: Model diagram. The components of the model shown in oval shapes are simulated populations of neurons that perform representations and transformations. The rectangular components are directly computed without neuron approximation. The connection between the *State and Action* population and the *Product* population calculates Eq. 3 according to the ideal state transition probabilities. The *Product* population performs an element-wise multiply between the transition probabilities and the Q-values stored in the environment. The connection from *Product* to the environment adds the results of that multiply. These last two steps together calculate Eq. 5.

The resulting Q-values need to be updated for both possible actions in the initial state. One possibility is to perform this in parallel (i.e., to have separate groups of neurons that perform this computation for each possible action). However, since this may be problematic if the number of actions grows to be large (or is unknown), we consider a serial strategy. Specifically, we have assumed the neural system considers actions one after the other over time. Although the actions are considered sequentially, all of the possible future states following those actions are considered in parallel.

Consistent with previous (non-neural) models of this task (Daw et al., 2011), the values of actions in the terminal, stage-two states are updated using a version of Q-learning:

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha r, \quad (6)$$

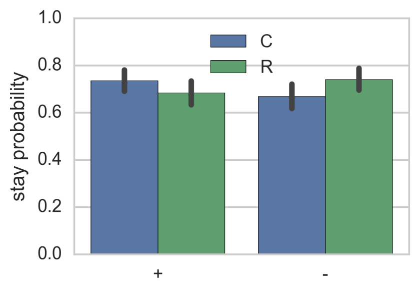
where α is a learning rate parameter and r is the immediate reward (Akam et al., 2015). This calculation is done directly, rather than with neurons.

Action selection is performed by approximating a softmax. That is, to determine the action performed in a given state, a small amount of random noise is added to the Q-values for all the actions in that state before selecting the action with the highest Q-value.

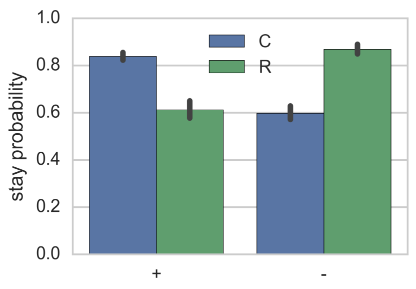
Method

We explore four parameters that influence the model's behaviour on the two-stage task:

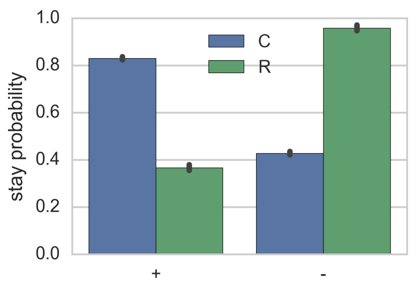
1. The **learning rate** α (from Eq. 6), which affects how much the Q-values of the terminal states are changed after receiving a reward.



(a) learning rate = 0.05



(b) learning rate = 0.3



(c) learning rate = 1

Figure 4: Examples of different stay probability behaviours for various learning rates with a Nengo seed of 1. Error bars are 95% confidence intervals.

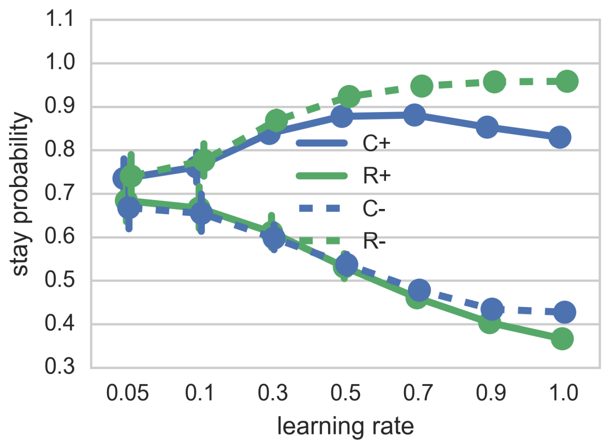


Figure 5: Stay probabilities of different learning rates. Error bars are 95% confidence intervals, and are sometimes smaller than data point markers.

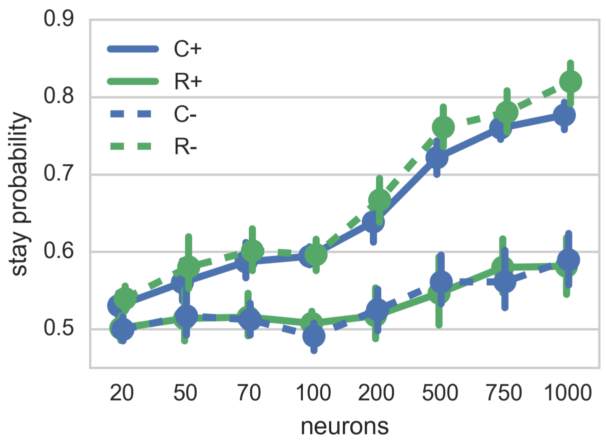


Figure 6: Stay probabilities of different numbers of neurons in the *State and Action* population. Model-based behaviour is clearly distinguishable with 100 or more neurons. Error bars are 95% confidence intervals.

2. The **number of neurons** in the *State and Action* and *Product* populations, which affects the accuracy of the representation and multiplication.
3. The random properties of the neurons used in the model are determined by Nengo according to a random seed. Different values of this seed produce different random distributions of neural tuning curves, so versions of the model instantiated with these different seeds can be thought of as different **individuals**.
4. The **time interval** between state transitions. The effect this is expected to have is that if the time interval is too short, the neurons will not have adequate time to compute the Q-values, and so the stay probability may be uniform in all rewarded (+) or unrewarded (−) and common (C) or rare (R) cases, since the agent is choosing actions based on essentially random information.

For each tested case, we run our simulations with twenty sessions of 10000 trials each. Each session is run with a different random seed for the environment, which determines the random behaviour of state transitions, the random noise in the action selection, and the random walks of reward probabilities.

Results

Learning rate

As shown in Figs. 4 and 5, as the learning rate α is increased, the effect of model-based reasoning is also increased; that is, the stay probabilities of the C+ and R− cases are increased, while the stay probabilities of the R+ and C− cases are decreased. When $\alpha = 1.0$, the most extreme example, there is no model-free learning; the Q-value of the terminal states is simply the reward (0 or 1) that was most recently received. In this situation, when calculating the Q-values of the initial state as in Eq. 4 or 5, the Q-values will either be exactly the

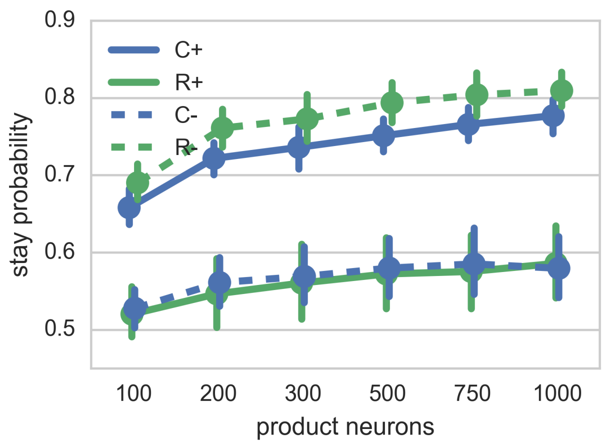


Figure 7: Stay probabilities of different numbers of neurons in the *Product* population. Error bars are 95% confidence intervals.

state transition probability, or 0. When α is a lower, for example $\alpha = 0.05$, much more emphasis is put on the learned values of the terminal states, and those values are learned much more slowly, and so they interfere with the model-based reasoning.

Number of neurons

In general, decreasing the number of neurons in the *State and Action* population has a similar effect to decreasing the learning rate: the stay probabilities of the C+ and R− cases decrease and the stay probabilities of the R+ and C− cases increase as the number of neurons decreases. This trend is shown in Fig. 6. Populations of at least 100 neurons were sufficient for producing clearly model-based stay probability behaviours. For most other simulations, a default value of 500 neurons was chosen because it produces a clear separation between C+, R− and R+, C− stay probabilities.

Increasing the number of neurons in the *Product* population above 200 per dimension did not produce any significant benefits, as shown in Fig. 7.

Individual

As shown in Fig. 8, there are large individual differences between different Nengo seeds. Many of them produce pure model-based stay probability plots (Fig. 8a), while some have a significant difference in stay probabilities between the R+ and C− cases that is reminiscent of human data (Fig. 8b), and in others, that significant difference is in the opposite direction to the human data (Fig. 8c). However, when averaged across individuals, the stay probabilities are characteristically model-based.

Time interval

As expected, it is necessary for the time interval between state transitions to be sufficiently long in order for the neurons to compute the Q-values. However, the individual differences discernible between different Nengo seeds are also dependent on the length of the time interval between state transitions; surprisingly, there is no apparent relationship between the length of the time interval and the stay probability behaviour. Three examples of stay probability behaviour with a Nengo seed of 1 are shown in Fig. 9. These can also be compared to Fig. 8b, which shows the same seed with a time interval of 50ms. Of particular interest is the discrepancy between Figs. 9b and 9c, since these time intervals have only a 10ms difference, yet show almost opposite stay probability behaviours.

Discussion

The core result of the research presented here is that, in general, the neural model of model-based reinforcement learning matches the expected results of a model-based agent. This is demonstrated by data aggregated across individual Nengo seeds, as well as particularly clearly by the trend produced by varying the learning rate α. The value of alpha that produced the greatest difference between the C+, R− and R+, C− stay probabilities (demonstrating a strong *model-based* effect) was $\alpha = 1.0$. This suggests that it may not be necessary for models of the two-stage task to use the model-free Q-learning component to estimate the values of terminal states (Eq. 6), since model-based stay probability behaviour can be produced when the values of the terminal states are taken to be the immediate reward.

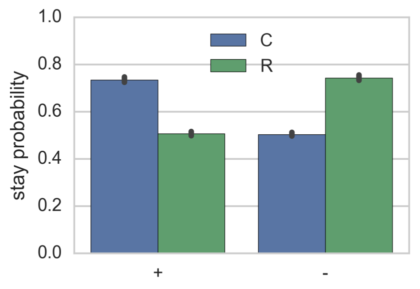
The effect of varying the number of neurons in the *State and Action* population is also as expected for a purely model-based agent. As the number of neurons increases, the representation of the current state and action is improved, which increases the likelihood of calculating the correct state transition probability as the input to the *Product* network.

The stay probability pattern reminiscent of human data reappeared with a number of Nengo seeds and time intervals. The individual differences between Nengo seeds demonstrates that the stay probability behaviour is surprisingly sensitive to the distributions of neurons. Future work will be done to further investigate this result.

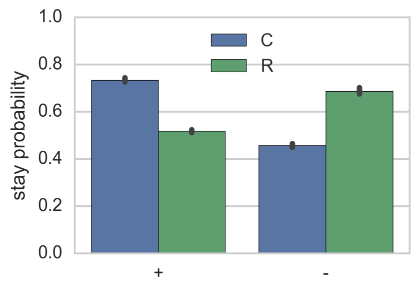
The most unforeseen result was that of the length of the time interval and its interaction with the Nengo seed. Increasing the length of the synaptic filter may eliminate this irregular effect; future work will investigate this and other possibilities.

Conclusion

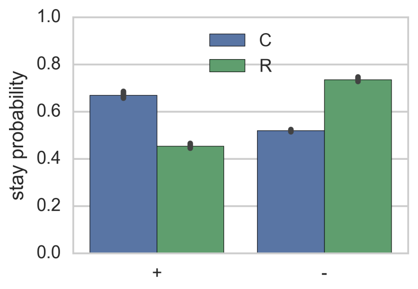
Our investigation of the effects of four parameters on the stay probability behaviour of a neural model of model-based reinforcement learning established that it typically performs as expected of a model-based agent. However, individual differences between certain parameter values demonstrated the model's sensitivity to the distribution of neural tuning curves and the time interval between state transitions.



(a) seed = 8

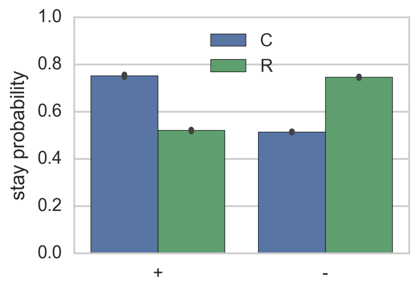


(b) seed = 1

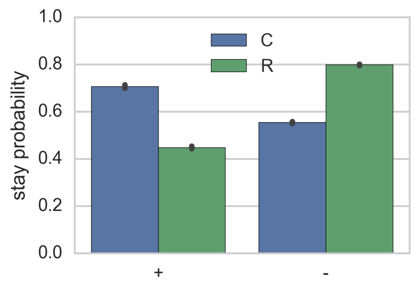


(c) seed = 7

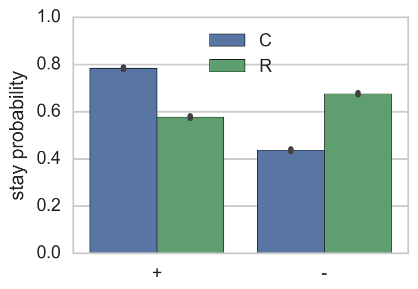
Figure 8: Examples of individual differences between Nengo seeds: (a) pure model-based stay probability behaviour, (b) stay probability behaviour suggestive of human data, and (c) stay probabilities opposite to those in (b). All trials were run with a time interval of 50ms. 95% confidence intervals are shown.



(a) time interval = 200ms



(b) time interval = 500ms



(c) time interval = 510ms

Figure 9: Examples of different behaviours for various time intervals with a Nengo seed of 1. Error bars are 95% confidence intervals.

Notes Supplemental material, including python scripts, is available at https://github.com/ctn-waterloo/cogsci17-rl/refactor.

Acknowledgments

This work was supported by CFI and OIT funding, the Canada Research Chairs program, NSERC Discovery grant 261453, ONR grant N000141310419, AFOSR grant FA8655-13-1-3084, and OGS and NSERC CGS-M graduate funding.

References

Akam, T., Costa, R., & Dayan, P. (2015). Simple plans or sophisticated habits? State, transition and learning interactions in the two-step task. *PLoS Computational Biology*, *11*(12).

Barto, A. G. (1995). *Adaptive critics and the basal ganglia* (J. C. Houk, J. Davis, & D. Beiser, Eds.). Cambridge, MA: MIT Press.

Bekolay, T., Bergstra, J., Hunsberger, E., DeWolf, T., Stewart, T. C., Rasmussen, D., ... Eliasmith, C. (2014). Nengo: A python tool for building large-scale functional brain models. *Frontiers in Neuroinformatics*, *7*(48).

Bellman, R. E. (1957). *Dynamic programming*. Princeton: Princeton University Press.

Daw, N. D., Gershman, S., Seymour, B., Dayan, P., & Dolan, R. J. (2011). Model-based influences on humans' choices and striatal prediction errors. *Neuron*, *69*, 1204–1215.

Daw, N. D., Niv, Y., & Dayan, P. (2005). Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control. *Nature Neuroscience*, *8*(12), 1704–1711.

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

Friedrich, J., & Lengyel, M. (2016). Goal-directed decision making with spiking neurons. *Journal of Neuroscience*, *36*(5), 1529–1546.

Glascher, J., Daw, N., Dayan, P., & O'Doherty, J. P. (2010). States versus rewards: Dissociable neural prediction error signals underlying model-based and model-free reinforcement learning. *Neuron*, *66*(4), 585–595.

Gosmann, J. (2015). *Precise multiplications with the nef* [Technical Report]. University of Waterloo, Waterloo, Ontario, Canada.

Lapicque, L. (1907). Recherches quantitatives sur l'excitation électrique des nerfs traitée comme une polarisation. *Journal de Physiologie et de Pathologie Générale*, *9*, 620–635.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. Cambridge: MIT Press.