A neural reinforcement learning model for tasks with unknown time delays

Daniel Rasmussen, Chris Eliasmith {drasmuss,celiasmith}@uwaterloo.ca
Centre for Theoretical Neuroscience, University of Waterloo <http://ctn.uwaterloo.ca>

University of Waterloo
CTN Waterloo
CNRG computational neuroscience research group
compneuro.uwaterloo.ca

Motivation/Problem

- **How can we perform reinforcement learning (RL) in a biologically plausible neural model?**
- **How do we extend this model to operate in environments with unknown/variable time delays between action selection, state transition, and reward?**

Reinforcement learning

Basic principles of RL that need to be incorporated into the model:

Q values

- Represent the immediate and future reward to be expected from selecting action *a* in state *s*

$$Q(s,a) = r(s,a) + \gamma Q(s',a')$$

TD learning

- Update Q values based on the difference between observed and expected value

$$\Delta Q(s,a) = \kappa\left[r(s,a) + \gamma Q(s',a') - Q(s,a)\right]$$

Semi-MDP (SMDP) TD learning

- Modify basic TD learning to incorporate the passage of time via semi-Markov framework

$$\Delta Q(s,a) = \kappa\left[\sum_{t=0}^{\tau-1}\gamma^t r(s,a,t) + \gamma^\tau Q(s',a') - Q(s,a)\right]$$

Neural Engineering Framework

Translate mathematical variables and computations into neural activities and connection weights:

Encoding (value → activities)

$$s_i(x(t)) = G_i\left[\alpha_i e_i x(t) + J_i^{bias}\right]$$

Decoding (activities → value)

$$\hat{x}(t) = \sum_i s_i(x(t)) d_i$$

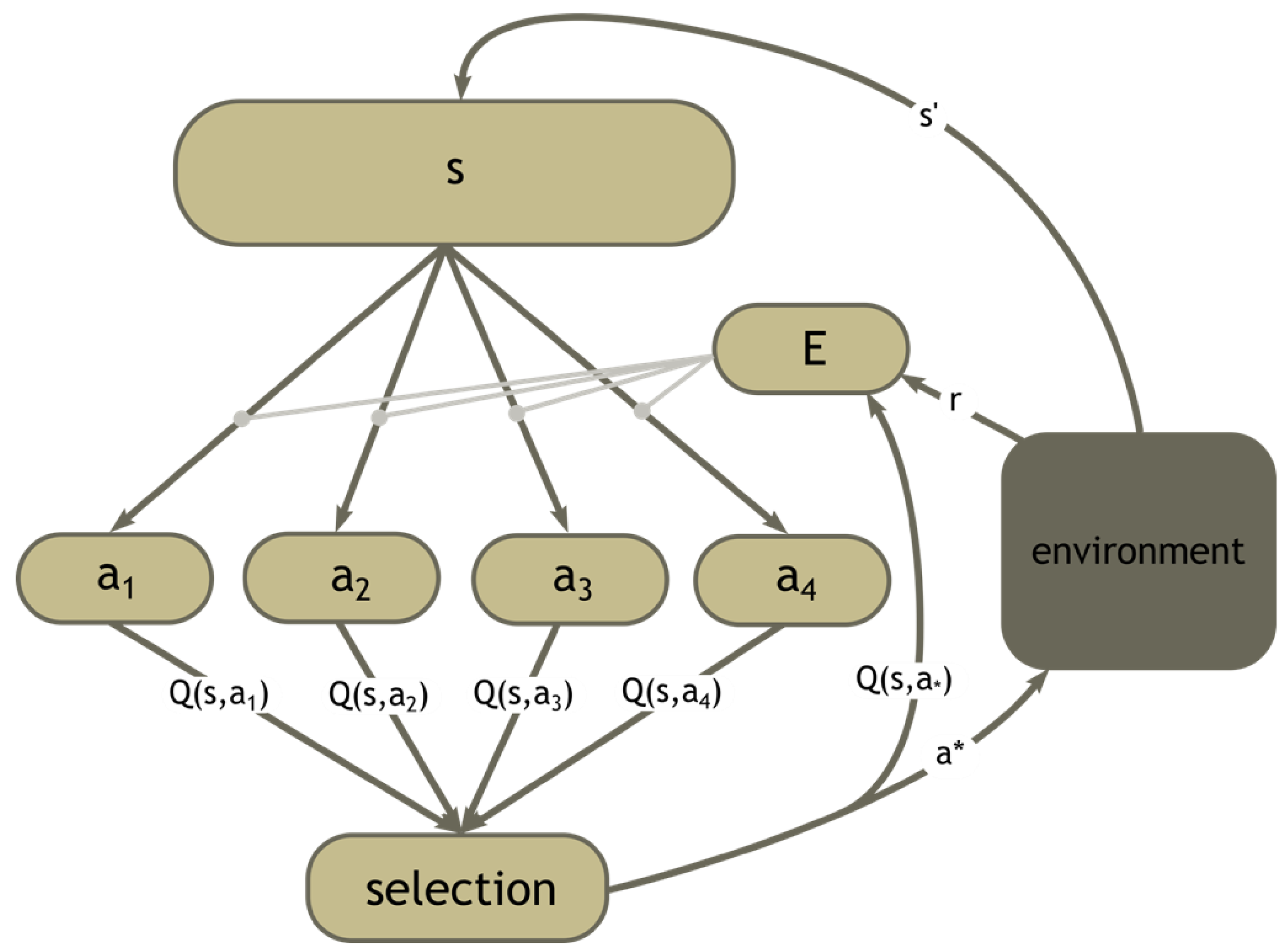
Analytically derived connection weights

$$\omega_{ij} = \alpha_j e_j C d_i$$

Error driven local learning rule

$$\Delta\omega_{ij} = \kappa\alpha_j e_j E s_i(x)$$

Model architecture



"s" population

- Represents the environmental state through distributed neural activities

"a" populations

- Compute the Q value of the respective actions based on output of "s" population

"selection" network

- Computes the highest valued action based on the outputs of the "a" populations (using a detailed model of the basal ganglia and thalamus)

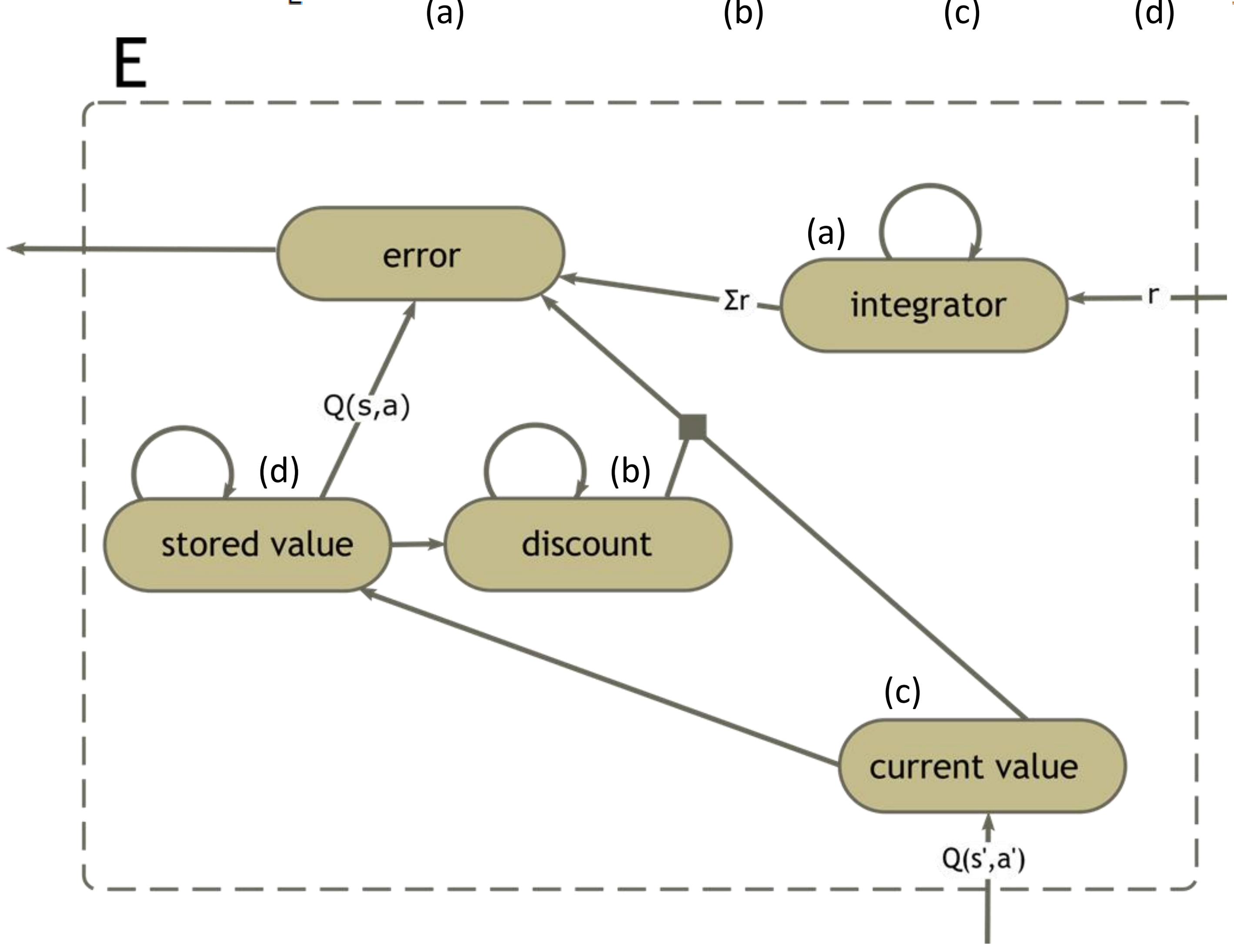
"environment"

- Can be any system that returns new state and reward based on action selected by agent

"E" network

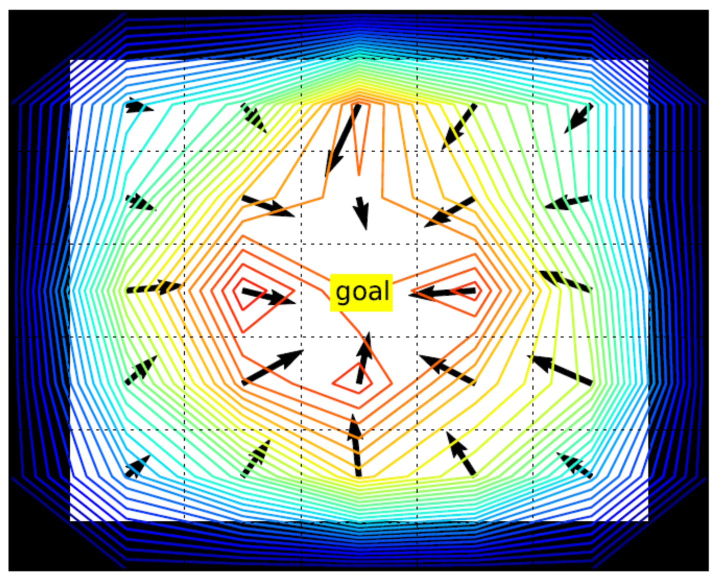
- Calculates the SMDP TD learning error signal (modified to work more accurately in a neural implementation)

$$\Delta Q(s,a) = \kappa\left[\underset{(a)}{\int_0^\tau r(s,a,t)\,dt} - \underset{(b)}{\int_0^\tau Q(s,a)\,dt} + \underset{(c)}{Q(s',a')} - \underset{(d)}{Q(s,a)}\right]$$

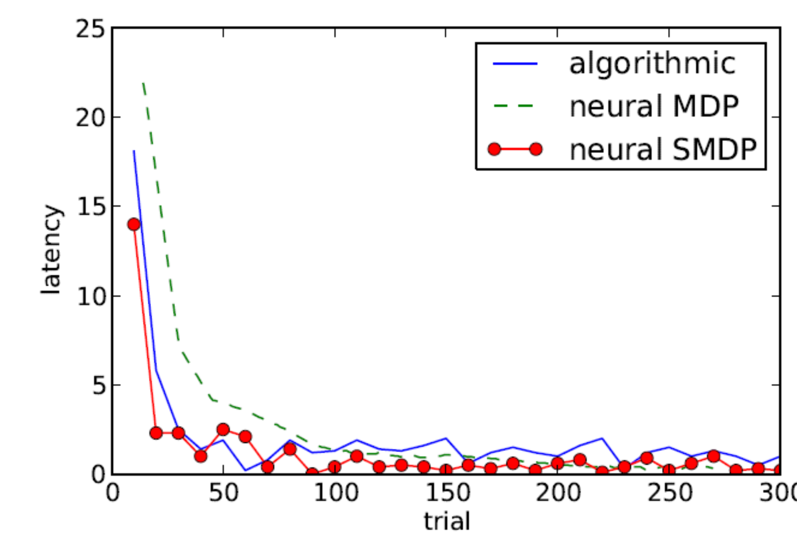


Results

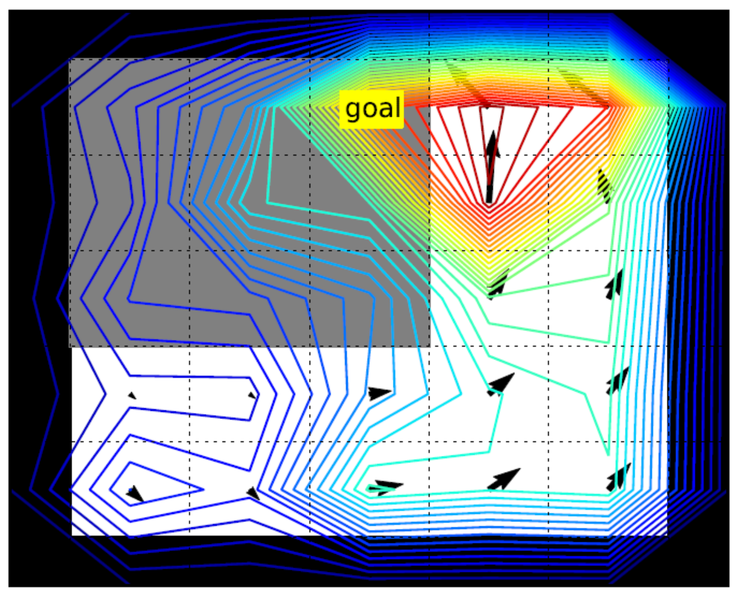
Successfully solves a watermaze-type navigation task where agent is placed in a random location in the world and must navigate to the goal. Only input is the current state (location of the agent) and a reward of 1 when in the goal state.



Learning time of the model (red line) compares well with other solutions.



Can use SMDP framework to incorporate time directly into the problem, e.g., adding "slow" areas (in grey) that agent learns to avoid.



Summary

Established new neural model able to perform reinforcement learning in a biologically plausible manner

- Operates in continuous time/space
- Computes TD error signal using only current reward as input

Works in SMDP environment

- Dealing with unknown/variable time delays
- Incorporating systematic delays into learned solution

Opens possibility of more advanced SMDP-based models (e.g., hierarchical reinforcement learning)