Information Processing and the Brain CW 2

Kheeran Naidu

December 2019

Question 1

The Reinforcement learning (RL) algorithm is based on the formal framework of Markov decision processes which is well defined in the notes. In CW2 the environment is represented by the 6×8 grid world which is well defined in the question sheet. In this finite world, the goal is for the agent to obtain the highest reward by following an optimal policy π (the actions to take at a particular state) of moving through the environment, which turns out to be the shortest path to the terminal state.

The algorithm is based on the expected reward of a given state s under a policy π. This *value function* is given by it's Bellman equation:

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)(r + \gamma v_\pi(s'), \forall s \in S, \tag{1}$$

where $\pi(a|s)$ is the probabilty of taking action a from state s under the policy π. With a complete knowledge of the behaviour of the environment and extreme computational costs, we can use dynamic programming (DP); where we have a process that assumes the values to be correct and updates the policies accordingly followed by another that assumes the policy to be correct and updates the values accordingly. This iterative process is called *bootstrapping* and converges to optimal values and policies when either process doesn't cause a change [7, Chapter 4].

However, without complete knowledge of the environment DP methods wouldn't work. Alternatively, Monte Carlo methods can learn directly from raw experience (i.e. from an agent moving in an environment). They work well but do not converge to optimal solutions. The trade-off comes in exporation vs explotation [7, Chapter 5].

More specifically, the RL algorithm using TD learning which was implemented for CW2 is a *policy iteration* that combines both methods. It uses bootstrapping and learns directly from raw experiences. Given some experience under a policy π, we update our estimate V of v_π for non-terminal states S_t using

$$V(S_t) \leftarrow V(S_t) + \alpha(\overbrace{R_{t+1} + \gamma V(S_{t+1}) - V(S_t)}^{\delta_t = \text{reward prediction error}}). \tag{2}$$

This updates the value at a learning rate of α with the TD-error (reward prediction error), δ_t - the difference between the estimate value of S_t and the better estimate $R_{t+1} + \gamma V(S_{t+1})$.

```
def update_value(state,policy,reward,value,W,alpha,discount):
    a = alpha
    d = discount
    value_new = np.copy(value)
    pos = state
    if valid_pos(pos,W,value.shape):
        pos_next = move_pos(pos,policy[pos],W,policy.shape)
        value_new[pos] = value[pos] + a*(reward[pos_next] + d*value[pos_next] - value[pos])
    return value_new
```

Figure 1: Code snipit of the one-step value update.

Now with our updated value estimate we update the policy for the state S_t to the best action a^* which moves to the state S_{a^*} such that $R(S_{a^*}) + \gamma V(S_{a^*})$ is maximised. This method doesn't include any exploration of the environment, so we use the $\epsilon-$greedy version of the algorithim. The only difference is that we update the policy to the best action a^* with probability $1-\epsilon$ and pick a random action otherwise. For the algorithm, $\epsilon = 1 - \frac{\text{current epoch}}{\text{total epochs}}$ to decrease exploration and increase exploitation as the epochs progress.

The main loop of the algorithm is shown in Figure 2. We begin in the current state and perform bootstrapping as described above. The agent moves to the next state according to the new policy and repeats until in the terminal state. At this point we repeat from the beginning with our new values and policy, incrementing the epoch.

```
alpha = 0.1
discount = 0.9
epochs = 500
for epoch in range(0,epochs):
    epsilon = 1 - epoch/epochs
    state = (5,0)
    while state != (2,2):
        value = update_value(state, policy, reward, value, W, alpha, discount)
        policy = update_policy(state, reward, policy, value, W, discount, epoch, epochs, epsilon)
        state = move_pos(state, policy[state], W, shape)
```

Figure 2: Code snippet of the main loop.

The results of the algorithm are shown in Figures 3, 4 and 5. The performance of the agent in learning the environment is calculated by its error given the current policy:

$$\text{error}_\pi = \sum_s [(\text{estimated shortest path from } s) - (\text{actual shortest path from } s)] \tag{3}$$

The algorithm converges to an optimal policy and as ϵ decreases the values of the state become less variable and eventually converge.

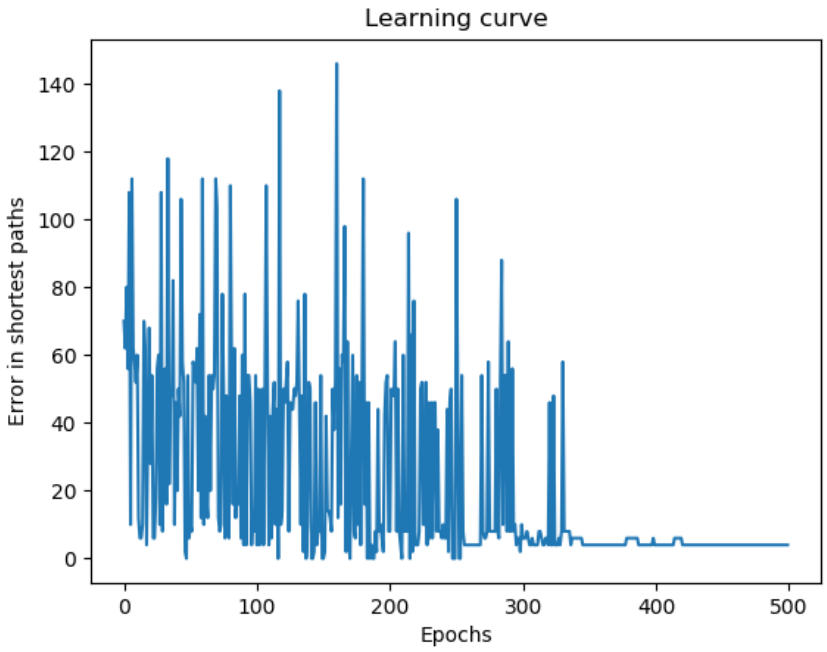

Figure 3: The learning curve of the algorithm based on error_π.

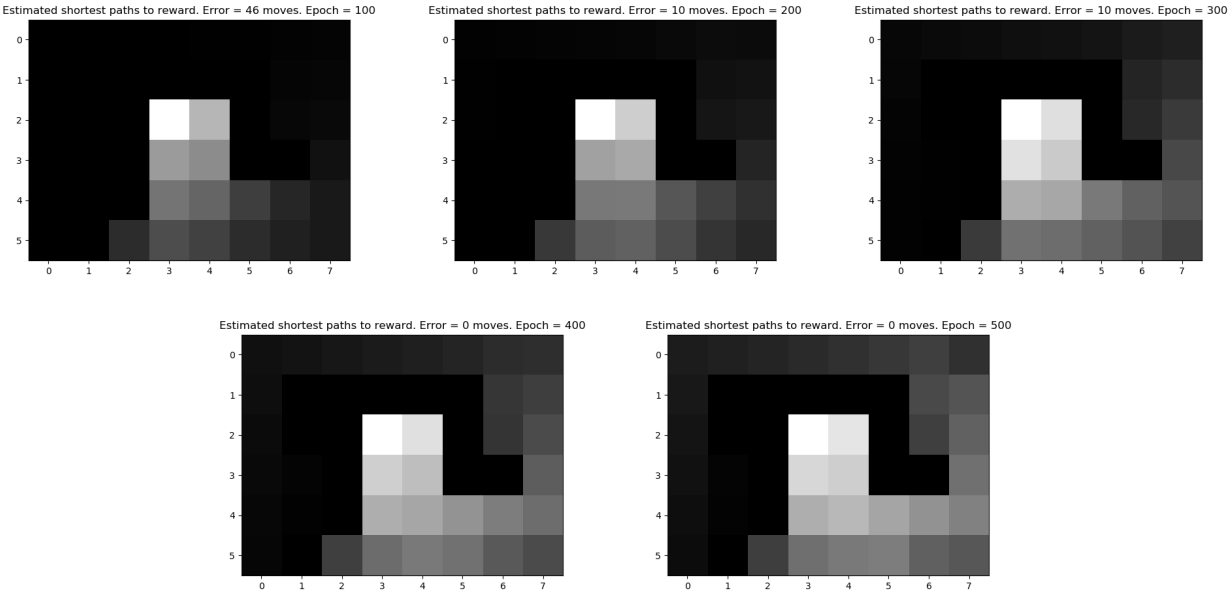

Figure 4: Heatmap of grid after every 100 epochs. This is a visualisation of the shortest paths where the policy is that the agent moves to the brightest adjacent tile.

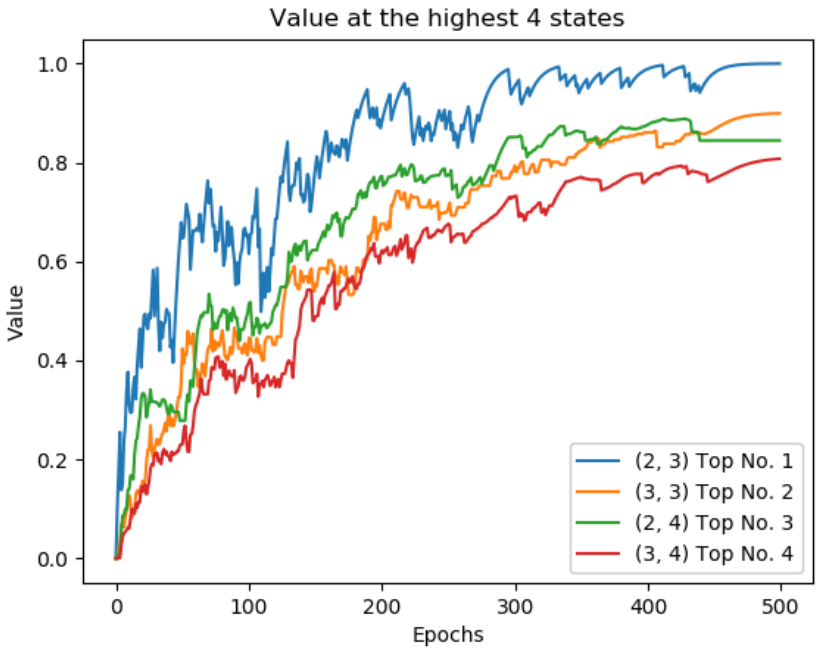

Figure 5: Change in the 4 highest valued states over the epochs.

Question 2

The RL algorithm is highly relatable to areas of the brain such as the dopaminergic systems [14] which code for unexpected reward. Most of the dopamine neurones of the brain originate from the substantia nigra or the ventral tegmental area of the midbrain [8].

Recent views discuss the significant role of the neurotransmitter dopamine in goal-directed behavior, cognition, attention, and reward [12]. Specifically in the case of rewards, a study on primates conducted in 1993 [1] and later analysed in 1997 [3] [6] developed an understanding of the function of the basal ganglia by a series of experiments on the activity of midbrain dopamine.

The experiments involved providing a monkey with a liquid reward upon learning a new task. During learning, the monkey's dopamine neurones responded to the liquid reward after completing the task. However, upon learning the task, the monkey's neurones respond to the a visual stimulus (a small light) before and cease to respond to the actual stimulus. This showed that the activity of dopamine neurones is affected not only by the present reward, but also the prediction of future rewards. The study also showed that when there was a delay in the reward after the visual stimulus, the dopamine neurones are depressed lower than their basal firing rates. This ties in very well with RL with TD learning. The TD-error δ_t (reward prediction error) of the TD learning rule behaves exactly like the dopamine neurones of the brain [9].

An analysis and comparison to the RL model [5] [10] showed that the input site of the basal ganglia is made of the striosome and the matrix. The striosome receives signals from the dopamine neurones of the substantia nigra pars conpacta (SNc) and the matrix receives receives output signals from the basal ganglia (substantia nigra pars reticulata and globus palidus). The model elaborates that the striosome is responsible for the reward at the current state whereas the matrix is responsible for future expected rewards.

Question 3

RL has a significant biological plausability over supervised learning as RL with TD learning has been shown to mimic the dopamine neurones in the mid-brain [9]. However, backpropagation of action potentials have been shown to not be equivalent to the backpropagation algorithm used in supervised and non-supervised learning [4].

RL has the ability to know when it is doing well, by virtue of the rewards obtained (or expected rewards). However, unsupervised learning has no way of understanding whether what has been learnt is truely correct.

Both supervised and unsupervised learning require tremendous amounts or difficult to obtain data before being able to make predictions. Additionally, once given a set of data, the goal is to find a good model of the data given. On the other hand, RL with TD learning learns on-the-go (online learning). It has the capability to explore an environment further (gather more data) to learn better. This mimics animals more closely as they are very rarely limited by a subset of data, and can explore the environment to learn more about it.

Question 4

In an environment with a very large state space, the agent would need to acumulate large amounts of data through exploration to make some reasonable policy for navigating the environment. We can overcome this by mapping the set of states as parameterized functions [7, Chapter 10] which would require less data to get an approximation of the optimal policy. Realistically, an environment with discrete states doesn't well represent a world with continuous states and so the environment of which an RL agent learns in is fundamentally different from the one animals learn in.

It has been shown that novel sensory stimuli induce similar activity to that of dopamine cells on unpredicted rewards [11]. The difference being that as the stimuli becomes familiar, activity diminishes. This was reasonded as novelty itself having rewarding characteristics [2] (intrinsic-motivation [10]). The algorithm we have used doesn't cater for the intrinsically-motivated rewards. Biologically, the intrinsic-motivation encourages exploration which we modelled by the epsilon-greedy mechanism ϵ. However, this isn't motivated by the novelty of a stimuli (or state in our case).

Sutton and Barto identify this shortcomming and note that an animal's reward signals are determined by processes within its brain that monitor both the animal's external and internal state [7]. Barto et al goes on to develop an *Intrinsically-Motivated RL* model [13] described in Figure 6.

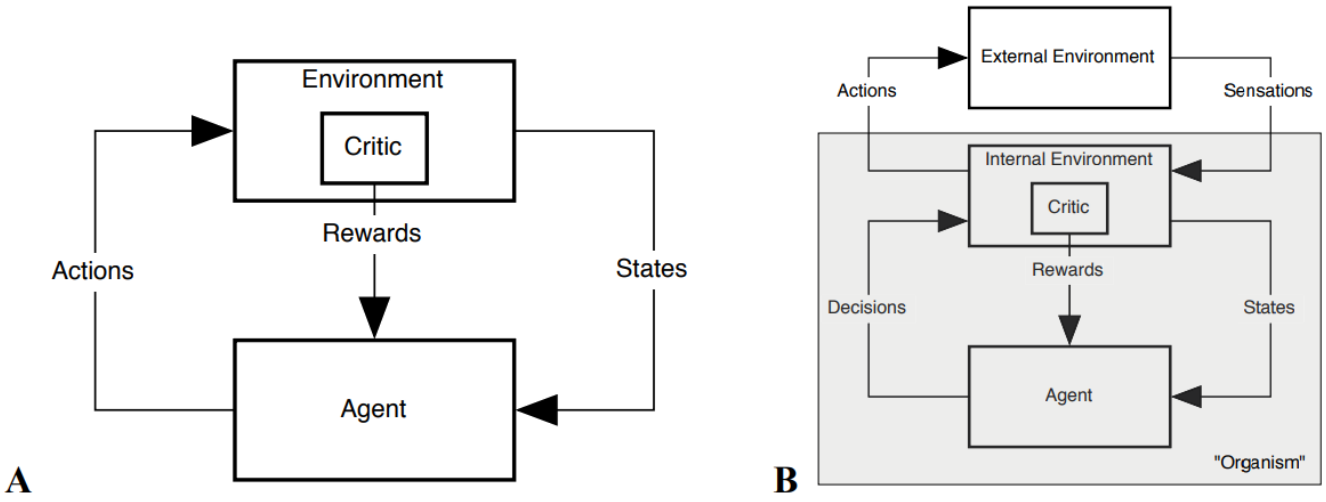

Figure 6: Agent-Environment Interaction in RL. **A**: The usual view. **B**: An elaboration where the internal environment of the model characterises the organism, in particular, it's motivational system *Sourced from [13].*

.

References

[1] Wolfram Schultz, Paul Apicella, and Tomas Ljungberg. "Responses of monkey dopamine neurons to reward and conditioned stimuli during successive steps of learning a delayed response task". In: *Journal of neuroscience* 13.3 (1993), pp. 900–913.

[2] Phil Reed, Chris Mitchell, and Tristan Nokes. "Intrinsic reinforcing properties of putatively neutral stimuli in an instrumental two-lever discrimination task". In: *Animal Learning & Behavior* 24.1 (1996), pp. 38–45.

[3] Wolfram Schultz, Peter Dayan, and P Read Montague. "A neural substrate of prediction and reward". In: *Science* 275.5306 (1997), pp. 1593–1599.

[4] Greg Stuart et al. "Action potential initiation and backpropagation in neurons of the mammalian CNS". In: *Trends in neurosciences* 20.3 (1997), pp. 125–131.

[5] Gregory S Berns and Terrence J Sejnowski. "A computational model of how the basal ganglia produce sequences". In: *Journal of cognitive neuroscience* 10.1 (1998), pp. 108–121.

[6] Wolfram Schultz. "Predictive reward signal of dopamine neurons". In: *Journal of neurophysiology* 80.1 (1998), pp. 1–27.

[7] Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*. Vol. 2. 4. MIT press Cambridge, 1998.

[8] Mahlon R DeLong. "The basal ganglia". In: *Principles of neural science* 4 (2000), pp. 647–659.

[9] Kenji Doya. "Complementary roles of basal ganglia and cerebellum in learning and motor control". In: *Current opinion in neurobiology* 10.6 (2000), pp. 732–739.

[10] Peter Dayan and Bernard W Balleine. "Reward, motivation, and reinforcement learning". In: *Neuron* 36.2 (2002), pp. 285–298.

[11] Sham Kakade and Peter Dayan. "Dopamine: generalization and bonuses". In: *Neural Networks* 15.4-6 (2002), pp. 549–559.

[12] Wolfram Schultz. "Getting formal with dopamine and reward". In: *Neuron* 36.2 (2002), pp. 241–263.

[13] Nuttapong Chentanez, Andrew G Barto, and Satinder P Singh. "Intrinsically motivated reinforcement learning". In: *Advances in neural information processing systems*. 2005, pp. 1281–1288.

[14] Pieter R Roelfsema and Anthony Holtmaat. "Control of synaptic plasticity in deep cortical networks". In: *Nature Reviews Neuroscience* 19.3 (2018), p. 166.