Information Processing and The Brain CW2

1. Temporal difference learning:

I implemented a path finding Agent using State-Action-Reward-State-Action (SARSA) alogrithms with greedy epsilon policy. (https://github.com/LinhPham123/TD-Learning)

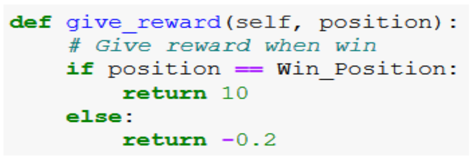
1.1 Basic knowledge:

Reinforcement learning is one of three basic machine learning paradigms. It is a process of making decisions consecutively and gaining information while learning, thus it can make correction to original prediction. Unlike supervised learning, it does not require labelled input/output pairs, instead, it acts bases on the reward that is interpreted from its previous actions in an environment. The main focus of reinforcement learning is to find a balance between exploration and exploitation.

Temporal difference learning is a model-free reinforcement learning method. It learns by bootstrapping from the current state value following a policy with no prior knowledge about the environment.

1.2 Implementation and methods used:

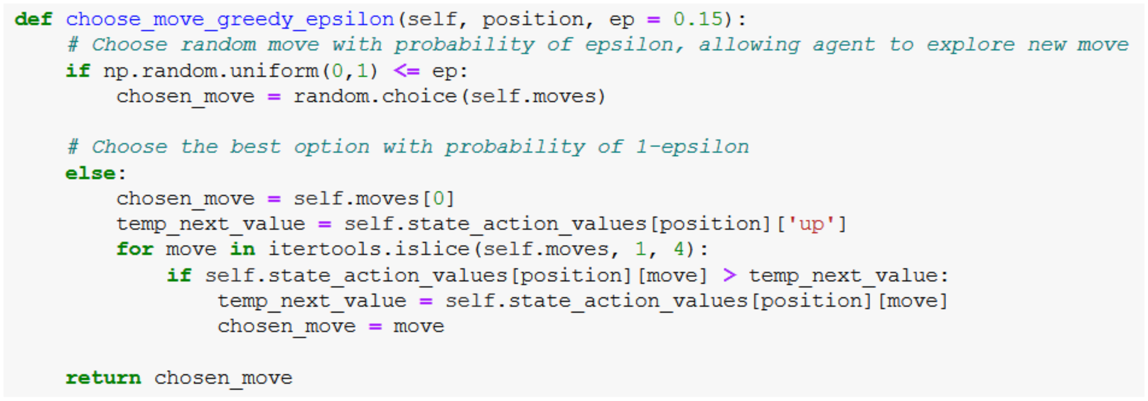
The Agent holds the pair values Q[S, A] of all states. At the start all values are set as 0.2 (e.g. (0,0): {“up”: 0.2, “down”: 0.2, “right”: 0.2, “left”: 0.2}). The Agent is rewarded 10 points when it reaches the end state and -0.2 for all other moves, so it is encouraged to find the end position. The values are updated after each move, including failed move such as hitting the walls or trying to go out of the world.



```
def give_reward(self, position):
    # Give reward when win
    if position == Win_Position:
        return 10
    else:
        return -0.2
```

Code snippet: Reward giving rule

With epsilon greedy policy, the Agent will choose a random action with epsilon probability or go for the action with maximum value with 1 – epsilon probability. This policy makes sure that the Agent will often choose the best move, but still have room to explore with new moves. Generally, epsilon should be high at the start to encourage exploration and lower toward the end for better exploitation. However, in this implementation, for the sake of simplicity, the epsilon is fixed at 0.15 for the whole learning process.



```
def choose_move_greedy_epsilon(self, position, ep = 0.15):
    # Choose random move with probability of epsilon, allowing agent to explore new move
    if np.random.uniform(0,1) <= ep:
        chosen_move = random.choice(self.moves)

    # Choose the best option with probability of 1-epsilon
    else:
        chosen_move = self.moves[0]
        temp_next_value = self.state_action_values[position]['up']
        for move in itertools.islice(self.moves, 1, 4):
            if self.state_action_values[position][move] > temp_next_value:
                temp_next_value = self.state_action_values[position][move]
                chosen_move = move

    return chosen_move
```

Code snippet: Greedy epsilon policy

SARSA is an on-policy temporal difference learning control method. It has the update rule:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \cdot \left(R_{t+1} + \gamma \cdot Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)\right)$$

Where $Q(S_t, A_t)$ is the state-actions values at state t

α is the learning rate which is important for convergence

γ $(0 \leq \gamma \leq 1)$ is the discount factor which determines whenever the Agent cares more about short term (smaller γ) or long term (larger γ) gains

R_{t+1} is the expected reward the Agent will get when it chooses to move to state t+1

```
position = self.State.current_state                                     # S_t
move = self.choose_move_greedy_epsilon(position)                        # A_t

self.move(move)                                                         # self.State.current_state = S_t+1
next_move = self.choose_move_greedy_epsilon(self.State.current_state)   # A_t+1

expected_reward = self.State.give_reward(self.State.current_state)      # R_t+1

#Q(S_t, A_t) = Q(S_t, A_t) + lr * (R_t+1 + df * Q(S_t+1, A_t+1) - Q(S_t, A_t))
reward = self.state_action_values[position][move] + self.lr \
            * (expected_reward + self.df * self.state_action_values[self.State.current_state][next_move] \
               - self.state_action_values[position][move])
self.state_action_values[position][move] = reward
```

Code snippet: SARSA in action

1.2 Results:

```
Value table for round 0

up     | up     | left   | left   | left   | left   | up     | right  |
down   | W      | W      | W      | W      | W      | down   | up     |
down   | W      | E      | left   | right  | W      | left   | up     |
down   | down   | W      | down   | right  | W      | W      | up     |
right  | down   | W      | left   | right  | left   | right  | down   |
S      | W      | left   | left   | down   | down   | up     | up     |


Value table for round 200

right  | right  | right  | right  | right  | right  | down   | down   |
up     | W      | W      | W      | W      | W      | right  | down   |
up     | W      | E      | left   | left   | W      | right  | down   |
up     | right  | W      | up     | left   | W      | W      | down   |
up     | down   | W      | up     | left   | left   | left   | left   |
S      | W      | right  | up     | up     | up     | left   | up     |


Final value table:

right  | right  | right  | right  | right  | right  | down   | down   |
up     | W      | W      | W      | W      | W      | right  | down   |
up     | W      | E      | left   | left   | W      | right  | down   |
up     | left   | W      | up     | left   | W      | W      | down   |
up     | left   | W      | up     | left   | left   | left   | left   |
S      | W      | right  | up     | up     | up     | left   | up     |
```

Figure 1: Agent's change in decisions through learning

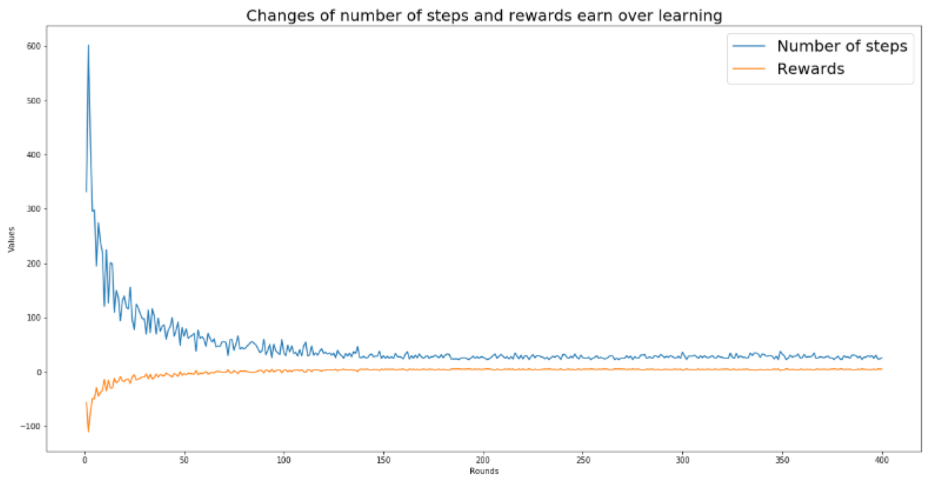


Figure 2: Number of moves needed to finish a round decreases and rewards earned increases through learning

2. Reinforcement learning and the brain:

The main idea behind reinforcement learning is the reward or punishment driven actions. These behaviours were first studied in detail by Ivan Pavlov in his famous experiments with his dogs. Given a stimulus (e.g. the sound of bell ringing) and a reward (e.g. food), after some repetitions, the brain will learn and response to the stimulus (e.g. dogs started to salivate when hearing the bell). He concluded that if a particular stimulus in the environment is presented when the reward is given, then the brain will associate that stimulus with the reward.

One study suggested that the phasic activity changes contributes to this behaviour by coding errors in the prediction of rewards, which was found by looking at electrophysiological recordings from dopamine neurons from rats, monkeys and humans [1]. Several other studies and experiments also confirmed the relation between dopamine and reinforcement learning [2][3][4]. Dopamine is a neuromodulator that is released from regions such as the Ventral Tegmental Area (VTA) whenever a pleasurable situation occurs or when an aversive stimulus is encountered. The targets of dopamine in human are the frontal cortex and ventral striatum which primarily moderates reward, reinforcement, motivational salience, stimulus-response learning and others executive functions [5][6].

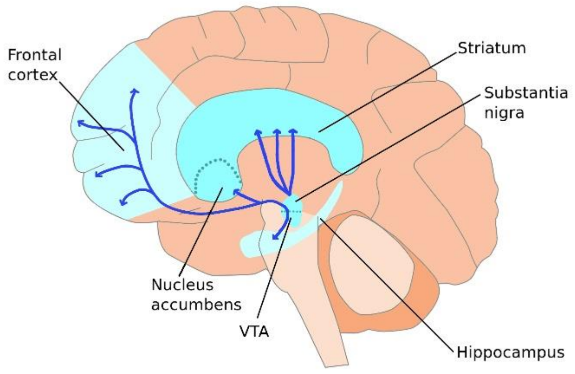


Figure 3: Dopamine pathways

In 1998, Wolfram Schultz conducted an experiment to determine the connection between temporal difference learning algorithms and dopamine by measuring the dopamine cells while training a monkey to associate a stimulus with a reward of juice. It is recorded that the firing rates of dopamine cells increased when the monkey received juice, which implies a difference in expected and actual rewards. The firing rate did not increase once the monkey was fully trained and was given the predicted juice. However, the firing rate would decrease below normal activation when the monkey was not given the predicted reward. This behaviour of the dopamine cells is similar to that of the error function in temporal difference learning [7].

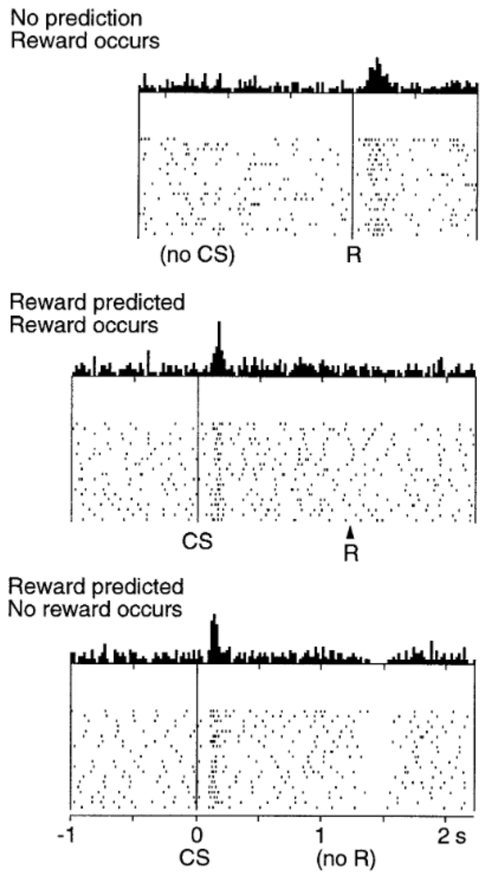


Figure 4: *firing rates of dopamine cells during the training [7]*

3. Advantages over the other two learning algorithms/paradigms:

Unlike supervised learning methods, temporal difference learning does not require the input and labelled output pairs, which in many situations, it would be too expensive or even impossible to provide (e.g. making move in chess). In additionally, the learning process of temporal difference learning is updated every one time step, and is not bounded by the provided labels. Thus, it gives the algorithms the ability to quickly adapt and revaluate the initial prediction based on its interaction with the environment.

Unsupervised learning methods require a huge amount of input data in order to process. Yet, they do not guarantee to give out a meaningful classification. Temporal difference learning, on the other hand, does not require a prior knowledge of the environment as it builds its own model while learning. Therefore, it can work well with limited input data.

4. Disadvantages over other two learning algorithms/paradigms:

Temporal difference learning, or reinforcement learning in general, is not feasible on tasks with enormous state spaces, as one would not be able to enumerate all possible states. Moreover, there could be too many possibilities so that implementing a practical solution is not realistic. This drawback, however, could be improved by combining both temporal difference learning and back propagation together. The reason is that, supervised learning has the ability to generalise learning across similar states. The resulting Agent would be able to flexibly learn over multiple time steps, as well as the structure of the input patterns, thus it can generalise the predictions over new states.

The exploration process of reinforcement learning is stochastic. Therefore, it is computational costly and time consuming. Contrastingly, supervised learning has expected outcomes, which means there is a guidance for every error signal in every neuron, thus, it will converge faster than other approaches.

Moreover, the paradigm is not a great model for real life situations where either not every action will result in a reward, or a reward is received once does not mean it can be earned every time.

Reference:

1. "*Neuroeconomics"* (Second Edition), 2014, Nathaniel D. Daw, Philippe N. Tobler.
2. *"Responses of monkey dopamine neurons to reward and conditioned stimuli during successive steps of learning a delayed response task"*, Schultz W, Apicella P, Ljungberg TJ Neurosci. 1993 Mar; 13(3):900-13.
3. *"Bee foraging in uncertain environments using predictive hebbian learning"*, Montague PR, Dayan P, Person C, Sejnowski TJ Nature. 1995 Oct 26; 377(6551):725-8.
4. "*Reinforcement Learning: An Introduction*", Sutton RS, Barto AG. Cambridge, MA: MIT Press; 1998.
5. *"The ins and outs of the striatum: Role in drug addiction", Yager LM, Garcia AF, Wunsch AM, Ferguson SM (August 2015). Neuroscience. 301: 529–541. doi:10.1016/j.neuroscience.2015.06.033. PMC 4523218. PMID 26116518.*
6. *"The neurocircuitry of illicit psychostimulant addiction: acute and chronic effects in humans", Taylor SB, Lewis CR, Olive MF (February 2013).*