Neural Networks and Biological Modeling

Professor Wulfram Gerstner
Laboratory of Computational Neuroscience

Question set 6

Exercise 1: Iterative update

Consider a reward-based learning system with an initial state s from which two actions a_1 and a_2 can be taken. Choosing an action a leads to a new state s', with a given probability $P^a_{s\to s'}$ and an associated reward $R^a_{s\to s'}$ (see Fig.1).

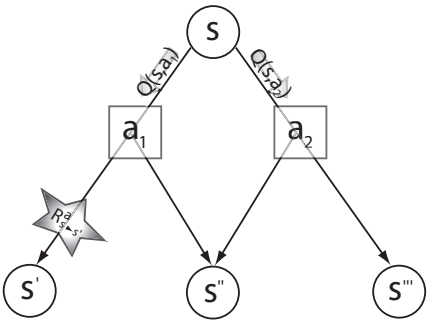


Figure 1.

1.1 Compare the empirical values of the Q-value for action a, $Q(s,a)$, evaluated either by averaging over k or over $k+1$ trials and use this to derive an iterative update rule of the form

$$\Delta Q(s,a) = \eta\,[r - Q(s,a)] \tag{1}$$

and identify the parameter η.

1.2 Give an inituitive explanation of this update rule.

Exercise 2

We now study the SARSA (state-action-reward-state-action) algorithm, with update rule

$$\Delta Q(s,a) = \eta\,[r - (Q(s,a) - Q(s',a'))]\ . \tag{2}$$

where s' and a' are the state and action subsequent to s and a. At each time step, the action chosen is the one with maximal expected reward, i.e.,

$$a^*_t = \arg\max_a Q_a(s,a)\,. \tag{3}$$

Consider a rat navigating in a linear track. The rat is initially placed at the upper end of the track (state s), with a food reward at the other end. This can be modeled as a linear sequence of states with a unique reward as the goal is reached. For each state, the possible actions are going up or going down (Fig.2). When the goal is reached, the rat is placed back in the initial position s and the exploration starts again.

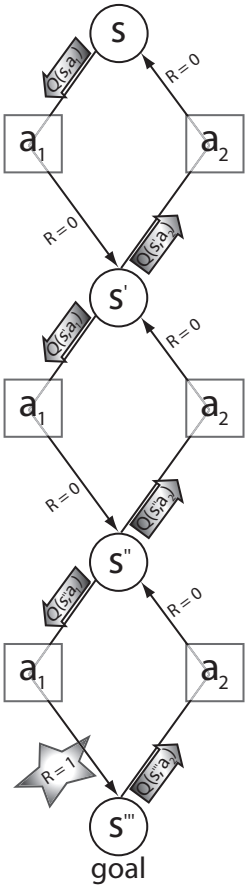


Figure 2.

2.1 Initialize all the Q-values at zero. How do the Q-values develop as the rat walks down the maze?

2.2 Calculate the Q-values after 3 complete trials. What happens to the learning speed if the number of states increases?

Exercise 3

3.1 Redo the previous exercise by adding an eligibility trace: for each state s and action a, a memory $e(s,a)$ is stored. At each time step, all the memories are updated as $e(s,a) = \lambda e(s,a)$, except for the memory corresponding to the current state s^* and action a^*, which is incremented:

$$e(s^*,a^*) = \lambda e(s^*,a^*) + 1\,. \tag{4}$$

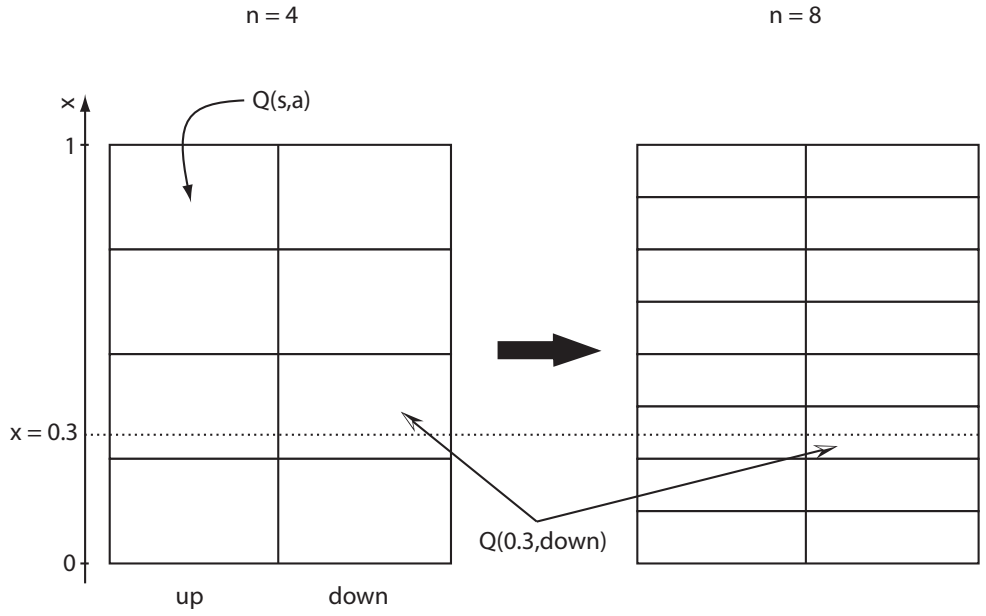
Now, unlike the case without eligibility treace, all Q-values are updated at each time step according to the rule

$$\forall(s,a) \quad \Delta Q(s,a) = \eta\,[r - (Q(s^*,a^*) - Q(s',a'))]\,e(s,a)\,. \tag{5}$$

where s^*, a^* are the current state and action, and s', a' are the immediately following state and action. How does the information propagation compare with that of the previous exercise?

3.2 (To be done at home) Suppose now that the actions are performed in a continuous 1-dimensional space $0 \le x \le 1$, with the target located at $x = 0$. Separate the interval into n equal steps $\Delta x = 1/n$. Start with $n = 4$, then consider $n = 8, 16\ldots$ Also suppose that the speed of movement $v = \Delta x/\Delta t$ is fixed independent of the choice of the discretization parameter n (e.g. for $n = 4$, $\Delta x = 1/4$ and $\Delta t = 1$ so that $v = 1/4$ and the neighbouring state is reached every time step).

How should we rescale the parameters Δt and λ, as $n = 4, 8, 16, \ldots$, in order that the "speed of information propagation" be constant? Consider for this the Q-value at $x = 0.3$ after 2 complete learning trials.



Exercise 4

Study the points discussed in the previous exercises in computer simulations, using the class `Gridworld` provided in the file `gridworld.py`. The class implements an agent that lives on an NxN grid, whose state can be characterized by two integer coordinates $(x, y) \in (0, ..., N-1) \times (0, ..., N-1)$. At each position, he can choose among 4 actions: 'Up', 'Down', 'Left' or 'Right'. The agent's task is to go to a previously unknown target position on the grid where he receives a reward. Moreover, he receives a smaller negative reward ('punishment') when bumping into walls. The agent learns according to the SARSA algorithm.

4.1 Size effects.

(A) Create 2 different gridworlds of size 5x5 and 10x10

```
>> import gridworld
>> gw1 = gridworld.Gridworld(5)
>> gw2 = gridworld.Gridworld(10)
```

and let the agents learn for 20 trials:

```
>> gw1.run(N_trials=20)
```

Look at a few trials using the method `visualize_trial()` for both networks. What is the qualitative difference and how would you explain it?

(B) Look at the navigation map of the agents, i.e., the action with the highest Q-value as a function of position

```
>> gw1.navigation_map()
```

How is the behavior of the agent reflected in the navigation maps? Compare the structure of the maps close to the target position and at larger distances.

(C) Quantify this difference by plotting the latencies (i.e., the time it takes the agent to reach the target position) as a function of trial number. This curve is often called the *learning curve*. To get smoother curves, let the agent take 20 runs on the same problem and plot the latency curve averaged over all runs:

```
gw1.run(N_trials=20,N_runs=20)
gw1.learning_curve()
```

Compare the latencies for the 2 gridworlds. What are the latencies that you would expect if the agent has discovered the optimal strategy and how do they compare to the latencies in the computer simulations?

4.2 Eligibility traces.

Create 2 different gridworlds of size 20x20, one of which learns with an eligibility trace and the other one without:

```
gw1 = gridworld.Gridworld(20)
gw2 = gridworld.Gridworld(20,lambda_eligibility=0.9)
```

Let the agents learn for 20 trials and a few runs (this might take a minute or so - why?). Compare the development of the latencies for the two agents. What do you observe and why?