ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Unsupervised and reinforcement learning in neural networks
Week 9

Lecturer: Marc-Oliver Gewaltig

Assistants: Dane Corneil
Berat Denizdurduran

RL in Continuous States

Exercise 1 (in class): Q-values for Continuous States

We approximate the state-action value function $Q(s, a)$ by a weighted sum of basis functions (BF):

$$Q(s, a) = \sum_j w_{aj} \Phi(s - s_j) \,,$$

where $\Phi(x)$ is the BF "shape", and the s_j's represent the centers of the BFs.
Calculate

$$\frac{dQ(s, a)}{dw_{aj}} \,,$$

the gradient of $Q(s, a)$ along w_{aj}.

Exercise 2: Eligibility trace

As seen in the exercise 8 (SARSA algorithm) from last week, it takes a long time to back-propagate the reward information through state space. The eligibility trace is introduced as a solution to this problem.
Reconsider the exercise from last week, but include 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 reduced by a factor λ: $e(s, a) = \lambda e(s, a)$, except for the memory corresponding to the current state s^* and action a^*, which is incremented:

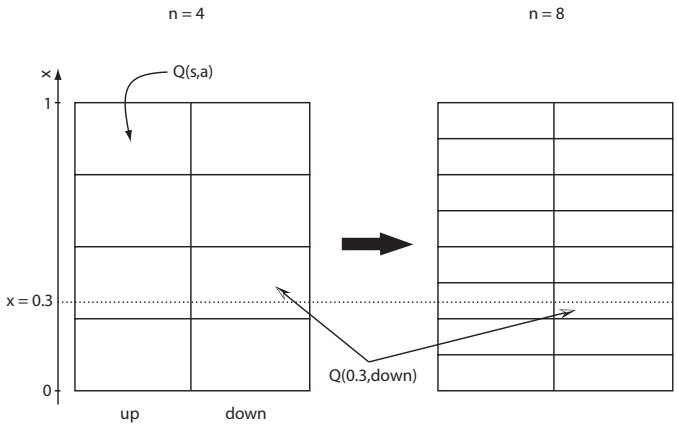
$$e(s^*, a^*) = \lambda e(s^*, a^*) + 1 \,. \tag{1}$$

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

$$\forall (s, a) \quad \Delta Q(s, a) = \eta \left[r - \big(Q(s^*, a^*) - Q(s', a') \big) \right] e(s, a) \,. \tag{2}$$

where s^*, a^* are the current state and action, and s', a' are the immediately following state and action. Does the information propagate more rapidly? (Assume that the agent goes straight down in the first trial and calculate the Q-values after two complete trials. Reset the eligibility trace to zero at the beginning of each trial.)

Exercise 3: Eligibility Trace for Continuous States



The left part of the figure above shows a different representation of last week's "linear track" exercise: the vertical divisions represent different states, and the two column correspond to the two possible actions available to the agent: go up or down. Each square represents a possible state-action combination, and thus a Q-value. (Note that the uppermost "up" action and the lowermost "down" action should be "greyed out": they are impossible. But this is not relevant to the rest of this exercise.)
Suppose now that the agent moves in a continuous 1-dimensional space $0 \leq x \leq 1$, with the target located at $x = 0$. Separate this state space into n equal bins of width $\Delta x = 1/n$. In each time step, the agent moves by one bin. Vary the discretization by varying n: $n = 4, 8, 16 \ldots$
How should we rescale the parameter Δt, so that the speed $v = \Delta x/\Delta t$ remains constant? How should we rescale λ, in order that the "speed of information propagation" in SARSA(λ) remains constant?

Exercise 4: Gradient-Based Learning of the Q-values

Assume again that the Q-values are expressed as a weighted sum of basis functions: $Q(s, a) = \sum_k w_k^a \Phi(s - s_k)$. Now consider the error function $E_t = \frac{1}{2}\delta_t^2$, where

$$\delta_t = r_t + \gamma \cdot Q(s', a') - Q(s, a)$$

is the reward prediction error.

1. Find a learning rule that minimizes the error function E_t by gradient decent. Consider the cases where (a) the actions a and a' are the same and (b) they are different.

2. How does the resulting learning rule relate to Hebbian learning rules that you know?

3. Argue (without calculations) what happens to the learning rule when the time interval between subsequent weight updates approaches zero.