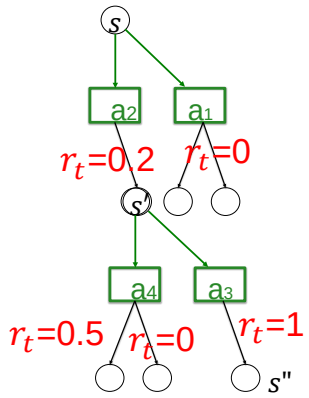
Artificial Neural Networks (Gerstner). Exercises for week 3

Variations of SARSA and continuous space

Exercise 1. Monte-Carlo Batch versus Bootstrap Batch

example trials:

1: **s,a2→** r=0.2,s',a4→ r=0
2: s',a3→ r=1
3: s',a4→ r=0
4: s',a3→ r=1
5: **s,a1** → r=0
6: s',a4→ r=0
7: s',a4→ r=0.5
8: s',a3→ r=1
9. **s,a2→** r=0.2,s',a4→ r=0.5
10. **s,a1** → r=0



The aim is to estimate Q-values on the graph as shown in the figure on top. We run 10 exploration trials with the rewards as indicated in the figure. Some trials start in state s others in state s'. Discount factor γ is equal to 1, and the policy is the random policy: $\pi(s,a) = 0.5$ for all state-action pairs. You use three different methods for estimating the Q-values: a. Batch Monte-Carlo, b. Batch-Expected SARSA, and c. Expected-SARSA.

a. Use Monte-Carlo in batch version to calculate the four Q-value (all combination of state-action pairs). To do so, evaluate for all state-action pairs the cumulative expected reward $R(s,a)$ with discount factor $\gamma = 1$ by averaging over all trials that either start at (s,a) or pass through (s,a).

b. Use Batch-Expected SARSA (= dynamic programming with an empirical component) to evaluate the estimated Q-values $Q(s,a)$ by exploiting the Bellman equation with discount factor $\gamma = 1$.

   Hints:

   (i) Start at the terminal state and work up backwards.

   (ii) The empirical Bellman equation is:

   $Q(s,a) = \langle r_t \rangle + \gamma \sum_{s'} P^{\text{emp},a}_{s\to s'} \sum_{a'} \pi(s',a') Q(s',a')$

   Here $\langle r_t \rangle$ is the empirical average over the immediate rewards for the combination (s,a) (over all trials), and $P^{\text{emp},a}_{s\to s'} = n(s'|s,a) / \sum_{s''} n(s''|s,a)$ is the empirical average of the transition matrix where $n(s''|s,a)$ is the number of trials in which you have taken action a in state s and landed in s''.

   (iii) Note that for action a_2, the branching ratio $P^{\text{emp},a}_{s\to s'}$ is unity, since there is only one possible landing state.

c. Use Expected-SARSA in the standard *online* version to evaluate the estimated Q-values $Q(s,a)$. To do so, use the optimal time-dependent learning rate η that we have seen in exercise 1 of last week. Since we have several states, you have to make $\eta(s,a)$ to depend

on count $n(s,a)$ for each state-action pair $\eta(s,a) = 1/n(s,a)$. You initialize all Q-values at zero and use discount factor $\gamma = 1$.

Hints:

(i) Convince yourself that after the first trial $Q(s,a_2)$ increases by 0.2.

(ii) Use the result of last week and exploit that the results of the specific time-dependent $\eta(s,a) = 1/n(s,a)$ in multi-arm bandit (and one-step horizon) tasks are in fact *equivalent* to batch. Thus, we know that after 8 trials, the Q-values $Q_8(s',a_3)$ and $Q_8(s',a_4)$ resulting from online Expected-SARSA gives are equivalent of a batch evaluation of the rewards after 8 trials.

(iii) Then analyze the update of $\Delta Q(s,a_2)$ after episode number 9. Remember that this is the *second* time that you play this action.

d. Calculate the exact Q-values for the random policy and assuming a balanced branching ratio where applicable.

e. Which of the algorithms performs best on this example?

Exercise 2. Eligibility traces

In week 2 in exercise 3, we applied the SARSA algorithm to the case of a linear track with actions 'up' and 'down'. We found that it takes a long time to propagate the reward information through state space. The eligibility trace is introduced as a solution to this problem.

Reconsider the linear maze from Fig 2. in exercise 3, 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

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

where s^*, a^* are the current state and action, and s', a' are the immediately following state and action.

We want to check whether the information about the reward propagates more rapidly. To find out, assume that the agent goes straight down in the first trial. In the second trial it uses a greedy policy. Calculate the Q-values after two complete trials and report the result.

Hint: Reset the eligibility trace to zero at the beginning of each trial.

Exercise 3. Q-values for continuous states

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

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

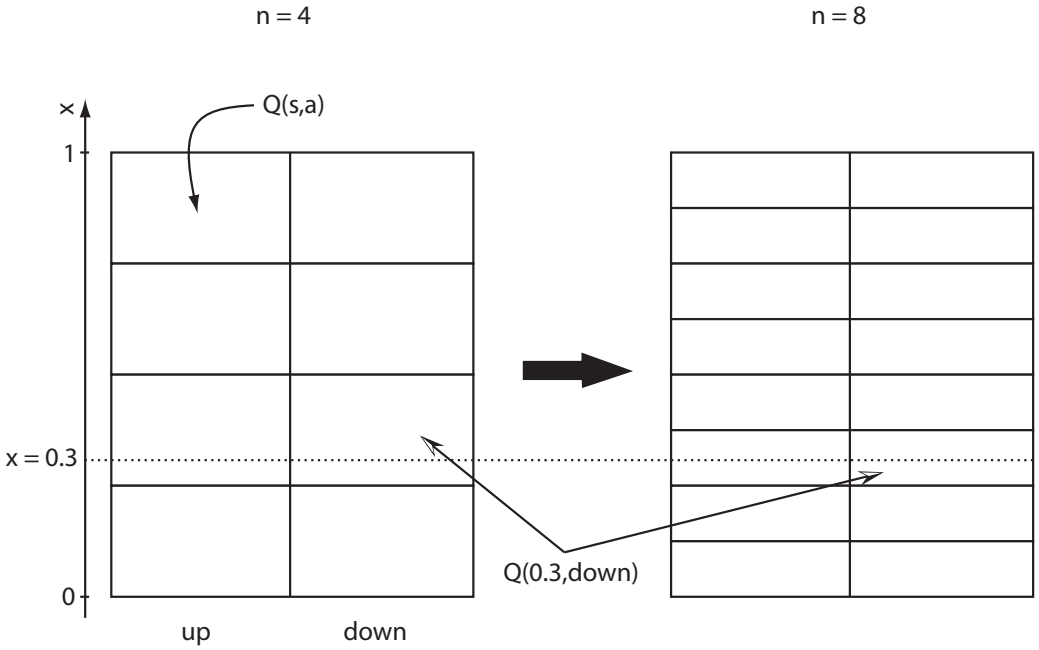
where $\Phi(x)$ is the BF "shape", and the s_j's represent the centers of the BFs.

Calculate

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

the gradient of $Q(s,a)$ along w_{ai} for a specific weight linking the basis function i to the action a.

Exercise 4. Eligibility traces in continuous space



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:
$n = 4, 8, 16 \ldots$

a. Suppose that one action (such as move down) corresponds to one time step Δt in 'real time'. How should we rescale the parameter Δt, so that the speed $v = \Delta x/\Delta t$ remains constant when we change the discretization?

b. We use an eligibility trace with decay parameter λ. How should we rescale λ, in order that the "speed of information propagation" in SARSA(λ) remains constant?

   Hint: Consider for example, the Q-value at $x = 0.5$ after 2 complete learning trials.

Exercise 5. Gradient-based learning of Q-values

Assume again that the Q-values are expressed as a weighted sum of 400 basis functions: $Q(s,a) = \sum_{k=1}^{400} w_a^k \Phi(s - s_k)$. For the moment the function Φ is arbitrary, but you may think of it as a Gaussian function. Note that s and s_k are usually vectors in $\mathbb{R}^N$ in this case. There are 3 different actions so that the total number of weights is 1200. Now consider the error function $E_t = \frac{1}{2}\delta_t^2$, where

$$\delta_t = r_t + \gamma \cdot Q(s', a') - Q(s, a) \tag{3}$$

is the reward prediction error. Our aim is to optimize $Q(s,a)$ by changing the parameters w.

a. Find a learning rule that minimizes the error function E_t by gradient decent. Consider the case where the actions a and a' are different.

   Write down the learning rule. How many weights need to be updated in each time step?

b. Find a learning rule that minimizes the error function E_t by gradient decent. Consider the case where the actions a and a' are the same.

   Write down the learning rule.

   Is there any difference to the case considered in a?

c. Suppose that the input space is two-dimensional and you discretize the input in 400 small square 'boxes' (i.e., 20×20). The basis function $\Phi(s - s_k)$ is now the indicator function: it has a value equal to one if the current state s is in 'box' k and zero otherwise.

   How do your results from (a) and (b) look like in this case?

d. The learning rule in c is very similar to standard SARSA. What is the difference?

e. Assume that $Q(s', a')$ in Equation 3 does not depend on the weights. How does the learning rule look like for a, b and c. How is your result related to standard discrete SARSA?

Exercise 6. Consistency condition for 3-step SARSA

In class we have seen the arguments leading to the error function arising from the consistency condition of Q-values.

$$E = 0.5 \sum [\delta_t]^2$$

with $\delta_t = r_t + \gamma Q(s', a') - Q(s, a)$ This specific consistency condition corresponds to 1-step SARSA.

Write down an analogous consistency condition for 3-step SARSA.