Artificial Neural Networks (Gerstner). Exercises for week 5

TD-learning and function approximation

Exercise 1. 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:

$$E = \frac{1}{2} \sum \delta_t^2$$

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.

Exercise 2. 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

$$\frac{\partial Q(s, a)}{\partial w_{\tilde{a}i}} \,,$$

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

Exercise 3. 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 this exercise 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

$$\delta_t = r_t + \gamma \cdot Q(s', a') - Q(s, a) \tag{1}$$

is the reward prediction error. Our aim is to optimize $Q(s, a)$ for all s, a by changing the parameters w. We consider $\eta \in [0, 1)$ as the learning rate.

a. Use the full gradient of the error function E_t and write down the learning rule based on gradient decent. Consider the case where the actions a and a' are different.

   How many weights need to be updated in each time step?

b. Use the full gradient of the error function E_t and write down the learning rule based on gradient decent. Consider the case where the actions a and a' are the same.

   Is there any difference to the case considered in (a)?

c. Repeat (a) and (b) by using the semi-gradient of the error function E_t. Do your answers change?

d. 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-c) look like in this case?

e. The learning rules in (d) are very similar to standard SARSA. What is the difference?

   Hint: Consider the difference between Full Gradient and Semi-gradient.

f. Assume that $Q(s', a')$ in Equation 1 does not depend on the weights. For example $Q(s', a')$ could be extracted from a separate neural network with its own parameters. How is your result in (a-c) related to standard SARSA? What do you conclude regarding the choice of semi-gradient versus full gradient? What do you conclude regarding the choice of Mnih et al. (2015) to model $Q(s', a')$ by a separate network with parameters that are kept fixed for some time?

Exercise 4. Inductive prior in reinforcement learning (from the final exam 2022)

We consider a 2-dimensional discrete environment with 16 states (Figure 1) plus one goal state where the agent receives a positive reward r. States are arranged in a triangular fashion in two dimensions. States are labeled as shown in the Figure 1 on the left. Available actions (Figure 1 on the right) are a_1 =up, a_2 =down, a_3 =right, a_4 =diagonally up right, a_5 =diagonally down right, a_6 =left (whenever these moves are possible). Returns are possible, e.g., the action up can be immediately followed by the action down.

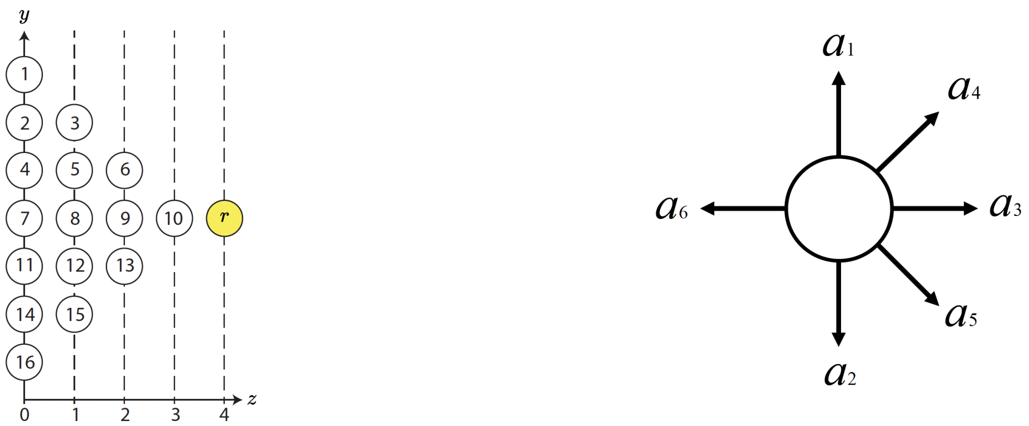

Figure 1: Figure for Exercise 4

Suppose that we use function approximation for

$$Q(a; X) = \sum_j w_{aj} x_j$$

with continuous state representation X with the following encoding scheme: Input is encoded in 18 dimensions $X = (x_1, x_2, ..., x_{16}, x_{17}, x_{18})$, where the first 16 entriesare 1-hot encoded discrete states; entry 17 is $x_{17} = 0.5 \cdot (z + 1)$ and $x_{18} = 0.1$ where z is the horizontal coordinate of the environment (Figure 1). Before the first episode, we initialize all weights at zero. During the first episode, we update Q-values using the Q-learning algorithm in continuous space derived with the semi-gradient method from the Q-learning error function. We consider $\eta \in [0, 1)$ as the learning rate and $\gamma \in [0, 1]$ as the discount factor.

a. Write down the quadratic loss function for 1-step Q-learning.

b. Using the semi-gradient update rule, what are the new weight values w_{ai} and Q-values $Q(s, a)$ for all 16 states and all actions at the end of the first episode? Write down all weights and Q-values that have changed.

c. In episode 2 you use a greedy policy in which ties are broken by random search. What is the probabilty p that the agent will choose a path with a minimal number of steps to the goal? Consider two initial states 7 and 11.

d. Is this behavior for episode 2 typical for 1-step Q-learning? Comment on your result in (c) in view of the no-free lunch theorem. (DO NOT write down the no-free lunch theorem, but use it in order to interpret your result.)

e. What can you say about the inductive prior of the variable x_{18}? To let you focus on the role of x_{18}, consider for a moment the representation $x_{17} = \alpha[z - \beta]$ with $\alpha = 0$ (instead $\alpha = 0.5$).

f. What can you say about the inductive prior of the variable x_{17}? To answer this question consider the representation $x_{17} = \alpha[z - \beta]$ and redo the calculations as in (b). Then compare parameters $\alpha = 0.5$ and $\beta = 2$ with parameters $\alpha = 0.5$ and $\beta = -1$.

What happens if the sing of α switches from $+1$ to -1?

g. What would be a great choice of functional representation for input x_{17} and x_{18} if you know that the reward is located at state 6 with coordinates $(z, y) = (2, 1)$?

Exercise 5. Review of TD algorithms 1[1]

You work with an implementation of 2-step SARSA and have doubts whether your algorithm performs correctly.

You have 2 possible actions from each state. You read-out the values after n episodes and find the following values:

$Q(1, a1) = 0$, $Q(2, a1) = 5$ $Q(3, a1) = 3$ $Q(4, a1) = 4$ $Q(5, a1) = 6$ $Q(6, a1) = 12$ $Q(7, a1) = 10$ $Q(8, a1) = 11$ $Q(9, a1) = 9$ $Q(10, a1) = 10$

$Q(1, a2) = 1$, $Q(2, a2) = 1$ $Q(3, a2) = 3$ $Q(4, a2) = 2$ $Q(5, a2) = 1$ $Q(6, a2) = 4$ $Q(7, a2) = 2$ $Q(8, a2) = 6$ $Q(9, a2) = 11$ $Q(10, a1) = 10$

You run one episode and observe the following sequence (state, action, reward)

$(1, a2, 1)$ $(2, a2, 1)$ $(3, a1, 0)$ $(5, a1, 4)$ $(6, a1, 1)$ $(8, a2, 1)$

What are the updates of 2-step SARSA that the algorithm should produce?

Exercise 6. Review of TD algorithms 2

Your friend proposes the following algorithm, using the pseudocode convention of Sutton and Barto.

Initialize $Q(s, a)$ = 0 for all $s \in \mathcal{S}, a \in \mathcal{A}$
Initialize π to be ε-greedy
Parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
All store and access operations (for S_t, A_t, and R_t) can take their index mod *4*

Repeat (for each episode):
  Initialize and store $S_0 \neq$ terminal
  Select and store an action $A_0 \sim \pi(\cdot|S_0)$
  $T \leftarrow$ 10000
  For $t = 0, 1, 2, \dots$:
  | If $t < T$, then:
  | Take action A_t
  | Observe and store the next reward as R_{t+1} and the next state as S_{t+1}
  | If S_{t+1} is terminal, then:
  | $T \leftarrow t + 1$
  | else:
  | Select and store an action $A_{t+1} \sim \pi(\cdot|S_{t+1})$
  | $\tau \leftarrow t -$ 3
  | If $\tau \geq 0$:
  | $X \leftarrow \sum_{i=\tau+1}^{\min(\tau+4,T)} \gamma^{i-\tau-1} R_i$
  | If $\tau + 4 < T$, then $X \leftarrow X + \gamma^4 Q(S_{\tau+4}\ A_{\tau+4})$
  | $Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha\,[X - Q(S_\tau, A_\tau)]$
  |
  Until $\tau = T - 1$

a. Is the algorithm On-Policy or Off-Policy?

   Answer: ....

b. What does the variable X represent?

   Answer ....

[1] Solving Exercise 5 is not nesscary. You can instead also run similar problems using simulations.

c. Is this algorithm novel, similar to, or equivalent to an existing algorithm?
   Answer (fill in/choose)
   This algorithm is identical/very similar to .... .
   There is no difference to the named algorithm/the main difference is ....
d. Is this algorithm a TD algorithm? What is the reason for your answer?
   Answer: Yes/No, because ....