Artificial Neural Networks (Gerstner). Solutions 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.

Solution:

The error function is $E = \frac{1}{2}\sum \delta_t^2$, but with a consistency condition $\delta_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 Q(s_{t+3}, a_{t+3}) - Q(s, a)$ where s_{t+3}, a_{t+3} are state and action three time steps after taking action a in state s. Explanation: the value $Q(s, a)$ must be explained by the (average of the) rewards in the next three steps plus the Q-value of the action three time steps after taking action a in state s. The averaging is taken implicitly by an online on-policy algorithm once the same sequence has been taken multiple times.

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 $\tilde{a}$.

Solution:

Using the definition of $Q(s, a)$ given, we find the gradient:

$$\frac{\partial Q(s, a)}{\partial w_{\tilde{a}j}} = \delta_{a\tilde{a}} \Phi(s - s_j)\,.$$

Therefore the direction of the gradient vector $(dQ(s, a)/dw_{aj})$ for $j = 1, \ldots, K$ is given by the magnitude of responses $\Phi(s - s_j)$ of all basis functions.

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?

Solution:

a. Let's start by computing the derivative of $Q(s, a)$ with respect to $w_{\tilde{k}}^{\tilde{a}}$ (we'll use this later):

$$\frac{\partial Q(s,a)}{\partial w_{\tilde{k}}^{\tilde{a}}} = \delta_{a\tilde{a}}\, \Phi(s - s_{\tilde{k}})\,,$$

   where $\delta_{a\tilde{a}}$ is the Kroneker, i.e., it is 1 if $a = \tilde{a}$, and 0 otherwise (not to be confused with δ_t).

   We then compute the gradient, i.e., the derivative of E_t with respect to $w_{\tilde{k}}^{\tilde{a}}$, using the chain rule a few times and the result above:

$$\begin{aligned}\frac{\partial E_t}{\partial w_{\tilde{k}}^{\tilde{a}}} &= \delta_t \left[\gamma \frac{\partial Q(s', a')}{\partial w_{\tilde{k}}^{\tilde{a}}} - \frac{\partial Q(s,a)}{\partial w_{\tilde{k}}^{\tilde{a}}}\right] \\ &= \delta_t \left[\gamma \delta_{a'\tilde{a}}\, \Phi(s' - s_{\tilde{k}}) - \delta_{a\tilde{a}}\, \Phi(s - s_{\tilde{k}})\right].\end{aligned}$$

   In gradient descent, we move the weights in the direction that *minimizes* the error, i.e.

$$\Delta w_{\tilde{k}}^{\tilde{a}} = -\eta \frac{\partial E_t}{\partial w_{\tilde{k}}^{\tilde{a}}} = \eta\, \delta_t \left[\delta_{a\tilde{a}}\, \Phi(s - s_{\tilde{k}}) - \gamma \delta_{a'\tilde{a}}\, \Phi(s' - s_{\tilde{k}})\right]$$

   $2 \cdot 400$ weights (for actions a and a') need to be updated in each step.

b. In the case where $a = a'$ (i.e., the action taken is the same in the two consecutive steps):

$$\Delta w_{\tilde{k}}^{\tilde{a}} = \eta\, \delta_t \left(\Phi(s - s_{\tilde{k}}) - \gamma \Phi(s' - s_{\tilde{k}})\right) \delta_{a\tilde{a}}.$$

   400 weights need to be updated.

c. When using semi-gradient, we assume that $Q(s', a')$ is fixed and independent of the weights. Hence, the semi-gradient of E_t with respect to $w_{\tilde{k}}^{\tilde{a}}$ is given by

$$\frac{\partial E_t}{\partial w_{\tilde{k}}^{\tilde{a}}} = \delta_t \left[\gamma \cdot 0 - \frac{\partial Q(s,a)}{\partial w_{\tilde{k}}^{\tilde{a}}}\right] = -\delta_t\, \Phi(s - s_{\tilde{k}}) \delta_{a\tilde{a}},$$

   which is importantly 0 for $\tilde{a} = a' \neq a$. Therefore, for both cases where $a = a'$ and $a \neq a'$, we have

$$\Delta w_{\tilde{k}}^{\tilde{a}} = \eta\, \delta_t\, \Phi(s - s_{\tilde{k}}) \delta_{a\tilde{a}}.$$

   400 weights need to be updated.

d. Showing box k by $\mathcal{B}_k$, we can write the Q-values as

$$Q(s,a) = \sum_{k=1}^{400} w_a^k I(s \in \mathcal{B}_k),$$

where $I(s \in \mathcal{B}_k)$ is the indicator function, i.e., is equal to 1 if $s \in \mathcal{B}_k$ and equal to 0 if $s \notin \mathcal{B}_k$.

The update rules based on the full gradient (a-b) can be written as (for part a, i.e., $a \neq a'$)

$$\Delta w_{\tilde{k}}^{\tilde{a}} = -\eta \frac{\partial E_t}{\partial w_{\tilde{k}}^{\tilde{a}}} = \eta\, \delta_t \, \left[\delta_{a\tilde{a}}\, I(s \in \mathcal{B}_{\tilde{k}}) - \gamma \delta_{a'\tilde{a}}\, I(s' \in \mathcal{B}_{\tilde{k}})\right]$$

and (for part b, i.e., $a = a'$)

$$\Delta w_{\tilde{k}}^{\tilde{a}} = \eta\, \delta_t \, \big(I(s \in \mathcal{B}_{\tilde{k}}) - \gamma I(s' \in \mathcal{B}_{\tilde{k}})\big)\, \delta_{a\tilde{a}}.$$

For the case of $a \neq a'$, 2 weights changes (for the boxes to which s and s' belong). For the case of $a = a'$, either 1 weight (if s and s' are in the same box) or 2 wieghts (if s and s' are in different boxes) change.

The update rule based on the semi-gradient (c) can be written as

$$\Delta w_{\tilde{k}}^{\tilde{a}} = \eta\, \delta_t\, I(s \in \mathcal{B}_{\tilde{k}}) \delta_{a\tilde{a}}.$$

Only 1 weight changes.

e. Let us define k as the index of the box for which we have $s \in \mathcal{B}_k$ and k' as the index of the box for which we have $s' \in \mathcal{B}_{k'}$. Using this notation, we have

$$Q(s,a) = w_a^k \quad \text{and} \quad Q(s',a') = w_{a'}^{k'}.$$

Therefore, the update rule based on the semi-gradient can be re-written as

identical to that of SARSA because it can be written as

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

which is identical to the update rule of SARSA.

The update rule based on the full gradient has an extra term given by $\gamma \Phi(s' - s_{\tilde{k}})$.

f. If $Q(s',a')$ is a fixed target that does not depend on the weights, then the full gradient and the semi-gradient are the same. This implies that the choice of the semi-gradient for the update rule is equivalent to the setting where $Q(s',a')$ is given by a separate neural network.

Hence, if $Q(s',a')$ is a fixed target, $\Delta w_{\tilde{k}}^{\tilde{a}}$ in (a-b) should be replaced by $\Delta w_{\tilde{k}}^{\tilde{a}}$ in (c). Hence, the update rules in (d) are all equivalent to the SARSA update rule.

As a result, the choice of Mnih et al. (2015) is equivalent to using semi-gradient with delayed update of $Q(s',a')$.

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.

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.

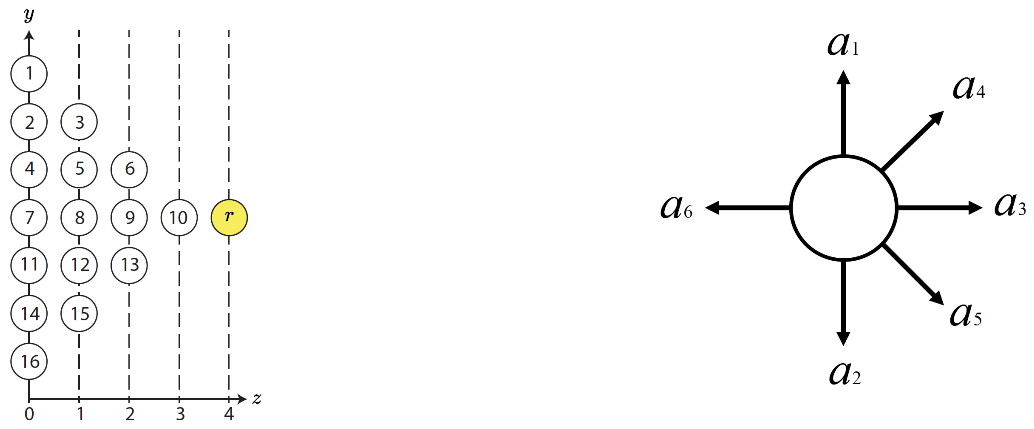

Figure 1: Figure for Exercise 4

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 $(z, y) = (2, 1)$?

Solution:

a. For the tuple $(X^t, a^t, r^{t+1}, X^{t+1})$, we have

$$\mathcal{L}(w) = \frac{1}{2}\left[\delta^t\right]^2$$

   with

$$\delta^t = r^{t+1} + \gamma \max_a Q(X^{t+1}, a) - Q(X^t, a^t).$$

b. Update of the weights for transition $(X^t, a^t, r^{t+1}, X^{t+1})$ is given by

$$\Delta w_{aj} = \eta \delta^t x_j^t \delta_{a,a^t}.$$

   Importantly, the only update happens after the tuple $(X^t = 10, a^t = a_3, r^{t+1} = r, X^{t+1} = \text{terminal})$ with $\delta^t = r$. The updated weights are given by

$$w_{aj} = \begin{cases} \eta r & \text{if } a = a_3 \text{ and } j = 10 \\ 2\eta r & \text{if } a = a_3 \text{ and } j = 17 \\ 0.1\eta r & \text{if } a = a_3 \text{ and } j = 18 \\ 0 & \text{otherwise} \end{cases}$$

and the updated Q-values by

$$Q(X, a) = \begin{cases} \eta r \left[\delta_{x_{10},1} + (z+1) + 0.01\right] & \text{if } a = a_3 \\ 0 & \text{otherwise.} \end{cases}$$

c. Starting from state 7, the agent with the greedy policy goes directly to the goal state, without any ties in the Q-values: $p = 1$.

For starting from state 11, the agent with the greedy policy goes directly to state 13, where is a tie between a_1, a_4, and a_6. To take the shortest, the agent needs to take a_6 with probability 1/3. Then, from state 10, it directly goes to the goal state: $p = \frac{1}{3}$.

d. No, it is a consequence of the particular functional form of Q function: it generalizes that the good actions are similar in all states ($x_{18} > 0$ and $x_{17} > 0$). This form is harmful in environment where the assumption is not satisfied (which is the price of the served lunch)!

e. When $x_{17} = 0$, we have

$$Q(X, a) = \begin{cases} \eta r \left[\delta_{x_{10},1} + 0.01\right] & \text{if } a = a_3 \\ 0 & \text{otherwise} \end{cases}$$

x_{18} adds a value to the rewarded actions in *all* states, so, in simple words, the inference prior of variable x_{18} is that the good action is the same for all states.

f. For $x_{17} = \alpha[z - \beta]$, we have

$$Q(X, a) = \begin{cases} \eta r \left[\delta_{x_10,1} + \alpha^2(3-\beta)(z-\beta) + 0.01\right] & \text{if } a = a_3 \\ 0 & \text{otherwise} \end{cases}$$

* In simple words, the inference prior of variable x_{17} is that the good action is similar among all states with $z < \beta$ (i.e., where $\alpha^2(3-\beta)(z-\beta) < 0$) but different from all states with $z > \beta$ (i.e., where $\alpha^2(3-\beta)(z-\beta) > 0$). Hence, for $\beta = 2$, agents starting from $X = 7$ will never take the direct path to the goal! For $\beta = -1$, the inference prior of variable x_{17} is qualitatively similar to that of x_{18}.

* The sign of α does not matter because only its squared appears in the updated weights.

g. One choice can be $x_{17} = (z-2)$ and $x_{18} = (y-1)$, but there are multiple good solutions with $f_1(z-2)$ and $f_2(y-1)$ for different functions f_1 and f_2

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?

Solution:

The update algorithm for 2-step SARSA is

$$\Delta Q(s_t, a_t) = \alpha(r_{t+1} + \gamma r_{t+2} + \gamma^2 Q(s_{t+2}, a_{t+2}) - Q(s_t, a_t))$$

with step size/learning rate α and discount factor γ. As a result, the update for the episode above should be

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

$$
\begin{aligned}
\Delta Q(1, a2) &= \alpha(1 + 1\gamma + 3\gamma^2 - 1) \\
\Delta Q(2, a2) &= \alpha(1 + 0\gamma + 6\gamma^2 - 1) \\
\Delta Q(3, a1) &= \alpha(0 + 4\gamma + 12\gamma^2 - 3) \\
\Delta Q(5, a1) &= \alpha(4 + 1\gamma + 6\gamma^2 - 6) \\
\Delta Q(6, a1) &= \alpha(1 + 1\gamma - 12) \\
\Delta Q(8, a2) &= \alpha(1 - 6).
\end{aligned}
$$

Here, we use the fact that no rewards can be received after the episode ends to truncate the summation. This can be thought of as a special "terminal" state at the end of each episode, that always transitions into itself with reward 0, and all Q–values equal to 0.

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 $t = 0, 1, 2, \ldots$:
  | 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 ....

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 ....

Solution:

a. The algorithm is On–Policy. In the third–to–last line, the value is bootstrapped using the Q–value estimate $Q(s_{t+4}, a_{t+4})$, i.e. the action that was taken in state s_{t+4} according to the agent's actual policy.

b. The variable X represents the 4–step truncated discounted returns. That is, X is a sample from the distribution over the returns that the agent can expect from taking action A_τ in state S_τ; the agent estimates the mean of this distribution with $Q(S_\tau, A_\tau)$.

   The agent gets this sample using the actual (discounted) rewards observed in the episode over the first 4 steps, plus an estimate of the average discounted returns from step 5 onwards (given by $\gamma^4 Q(S_{\tau+4}, A_{\tau+4})$).

c. The algorithm is equivalent to 4–step SARSA, which itself is very similar to the more commonly used 1–step SARSA.

d. The algorithm is a TD algorithm because it uses bootstrapping (updating estimates from other, later estimates) to estimate the target (the Q–value function).