Last Name ..... First Name.....

Artificial Neural Networks / Reinforcement Learning: Exam June 2024 (with solutions)

- Keep your bag next to your chair, but do not open it during the exam.
- Write your name in legible letters on top of this page.
- The exam lasts 180 min.
- Write **all** your answers in a legible way on the exam (no extra sheets).
- No documentation is allowed (no textbook, no slides), except **one page A5 of handwritten notes, doublesided**.
- No calculator is allowed.
- Have your student card displayed in front of you on your desk.
- **Check that your exam has 13 pages.**

Evaluation:

1. ..... / 7 pts (Section 1, Debugging a RL algorithm)
2. ..... / 8 pts (Section 2, Policy Gradient)
3. ..... / 10 pts (Section 3, 3-step Deep SARSA)
4. ..... / 10 pts (Section 4, Inference Prior)

Total: ..... / 35 pts

Definitions and notations

RL stands for Reinforcement Learning.

Bold face symbols refer to vectors, normal face to a single component or a single input/output. Unless noted otherwise, the input is $N-$dimensional: $\mathbf{x}^\mu \in R^N$.

Throughout the exam, we denote the learning rate by η. The symbol a refers to an action; the symbols r to a reward; the symbol s to a discrete state; and the symbol γ to a discount rate.

If the state space is continuous then states are also written as $\mathbf{x}$.

If algorithms are implemented as neural networks, the parameters are the weights w_{nm} from neuron m to neuron n.

How to give answers

The remaining sections involve calculations. **Please write the answers in the space provided for that purpose.**

We also provide some free space for calculations. We will not look at these parts for grading. You can ask for extra scratch paper. We will not look at the scratch paper.

1 Debugging a RL algorithm (7 points)

For debugging of an RL algorithm, you run an agent in a small environment consisting of 6 states with 2 possible actions from each of the 6 states. After running the agent for n episodes, the table of Q-values is as is shown below. For example, we have $Q(5, a1) = 5$.

actions \ states	1	2	3	4	5	6
$a1$	1	2	3	4	5	6
$a2$	3	2	0	2	0	2

Your policy is computed as

$$\pi(a_i|s) = \frac{Q(s, a_i)}{Q(s, a_1) + Q(s, a_2)} \tag{1}$$

You run one additional episode and observe the following sequence of several steps, denoted as (s, a, r) =(state, chosen action, observed reward triggered by transition)

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

After step 6 the episode ended.

(i) **Expected SARSA**: You work with an implementation of 1-step online expected SARSA on your computer, using a discount factor of $\gamma = 1$ and a learning rate of $\eta = 0.1$. Perform the first 5 updates of your algorithm. After the 5 updates, which Q-values have changed and what is the new value? Fill in the blanks. *Example: update 8:* $Q(9, a3) = 3.2$.

Update 1: Q(1,a2) = 3.1

Update 2: Q(2,a2) = 2.1

Update 3: Q(3,a1) = 3 $\frac{1}{3}$

Update 4: Q(4,a2) = 2.2

Update 5: Q(5,a1) = 5.2

number of points: ...../ 2

(ii) Assume now that you work instead with an implementation of **1-step Q-learning** on your computer, using a discount factor of $\gamma = 1$ and a learning rate of $\eta = 0.1$. Which of the 5 updates would be equivalent to the updates from part (i)? *Hint: You do not need to compute the updates. Example: The updates that would be the same as in part (i) are updates 8, 9 and 11.*

The update(s) that would be the same as in part (i) are update(s) Updates 1, 2, 4

number of points: ...../ 2

(iii) Assume now that you work with an implementation of **1-step regular SARSA**, using a discount factor of $\gamma = 1$ and a learning rate of $\eta = 0.1$. Which of the 5 updates would be equivalent to the updates from part (i)?

The update(s) that would be the same as in part (i) are update(s) Updates 1, 2, 4

number of points: ...../ 2

(iv) For which of the five updates would regular SARSA (part iii) and Q-learning (part ii) be **different**?

The update(s) that would be different between regular SARSA and Q-learning are update(s) Updates 3, 5

number of points: ...../ 1

2 Policy Gradient (8 points)

The agent operates in an environment with states s and actions a. At each time step, a new state s is drawn from a probability distribution $P(s)$. Upon observing a state s, the agent selects an action a following a policy $\pi_{\mathbf{w}}(a|s)$, parameterised by parameters w_{nm} arranged in a matrix $\mathbf{w}$, and receives deterministic rewards as $r(s,a)$.

(i) Write down the expected reward $E[r(s,a)|\mathbf{w}]$ for the agent:

$E[r(s,a)|\mathbf{w}] = \sum_s P(s) \sum_a \pi_{\mathbf{w}}(a|s) \cdot r(s,a)$

number of points: ...../ 1

(ii) Compute the gradient $\nabla_{\mathbf{w}} E[r(s,a)|\mathbf{w}]$ while keeping a valid statistical weight in the formula.

$$\begin{aligned}\nabla_{\mathbf{w}} E[r(s,a)|\mathbf{w}] &= \sum_s \sum_a P(s) \cdot \pi_{\mathbf{w}}(a|s) \cdot \nabla_{\mathbf{w}} \ln(\pi_{\mathbf{w}}(a|s)) \cdot r(s,a) \\ &= E[r(s,a) \nabla_{\mathbf{w}} \ln(\pi_{\mathbf{w}}(a|s))]\end{aligned}$$

number of points: ...../ 2

(iii) From your answer in (ii), derive an online update rule for weight w_{nm}:

$\Delta w_{nm} = \eta r(s,a) \frac{\partial \ln \pi_{\mathbf{w}}(a|s)}{\partial w_{nm}}$

number of points: ...../ 1

(iv) Show that in expectation your online version in (iii) leads back to the correct gradient of $E[r(s,a)|\mathbf{w}]$:

$$\begin{aligned}E\left[r(s,a)\frac{\partial \ln \pi_{\mathbf{w}}(a|s)}{\partial w_{nm}}\right] &= \int p(s) \sum_a \pi_{\mathbf{w}}(a|s) r(s,a) \frac{\partial \ln \pi_{\mathbf{w}}(a|s)}{\partial w_{nm}} ds \\ &= \int p(s) \sum_a \pi_{\mathbf{w}}(a|s) r(s,a) \frac{1}{\pi_{\mathbf{w}}(a|s)} \frac{\partial \pi_{\mathbf{w}}(a|s)}{\partial w_{nm}} ds \\ &= \int p(s) \sum_a r(s,a) \frac{\partial \pi_{\mathbf{w}}(a|s)}{\partial w_{nm}} ds \\ &= \frac{\partial E[r(s,a)]}{\partial w_{nm}}\end{aligned}$$

number of points: ...../ 1

(v) We now consider that the policy is given as follows:

$$\pi_{\mathbf{w}}(a=i|s) = \frac{e^{\sum_{k=1}^{K} w_{ik} y_k}}{\sum_j e^{\sum_{k=1}^{K} w_{jk} y_k}} \tag{2}$$

where $\mathbf{w}_i$ is the weight vector for action $a = i$, and $y_k = f_k(s)$ with $1 \leq k \leq K$ is a set of basis functions.

Using Eq. (2), find an elegant and compact formula for the online update of the weight w_{nm}:

$\Delta w_{nm} = \eta y_m(\delta_{i,n} - \pi_{\mathbf{w}}(a = i|s))r(s, a = i)$

number of points: ...../ 1.5

(vi) Relate your result from (v) to 3-factor learning rules by specifying the three factors and giving their interpretation in the context of learning in the brain. Fill in the blanks (the order of factors does not matter).

The 1st factor is y_m. It can be interpreted as presynaptic input .

The 2nd factor is $\delta_{i,n} - \pi_{\mathbf{w}}(a = i|s)$. It can be interpreted as postsynaptic activity.

The 3rd factor is $r(s, a = i)$. It can be interpreted as the global third factor.

number of points: ...../ 1.5

3 3-step Deep SARSA (10 points)

We approximate Q-values by function approximation in a deep network with weights $w_{ij}^{(k)}$. The input $\mathbf{x}$ is continuous.

(i) Write down a "reasonable" loss function resulting from the consistency condition of the Bellman equation for the case of a deep network implementing **3-step SARSA**.

Batch: Loss $L_{bt} = \langle \frac{1}{2}\delta^2 \rangle$

Online: Loss $L_{on} = \frac{1}{2}\delta^2$

with $\delta = (r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 Q_{\mathbf{w}}(\mathbf{x}_{t+3}, a_{t+3}) - Q_{\mathbf{w}}(\mathbf{x}_t, a_t))$

number of points: ...../ 1

(ii) What is the difference between **full gradient** and **semi-gradient**?

Full gradient takes the derivative of the full loss with respect to the weights $\mathbf{w}$, semi-gradient considers $Q(\mathbf{x}_{t+3}, a_{t+3})$ as fixed.

number of points: ...../ 0.5

(iii) Write down the online update rule derived from point (i) using **semi-gradient**:

Update: $\Delta w_{ij} = \eta\delta \frac{\partial Q_{\mathbf{w}}}{\partial w_{ij}}(\mathbf{x}_t, a_t)$

number of points: ...../ 1

In class, we discussed the interest of using two separate networks: a fast network Q_{fast} and a slow network Q_{slow}.

(iv) What do "fast" and "slow" refer to?

Fast and slow refer to the rate of update of the network's parameters; Q_{fast} is updated in every step, whereas Q_{slow} is updated only every C time steps and is often used as the target in the TD error.

number of points: ...../ 0.5

(v) Analogous to part (i), write down the loss function for a deep network implementing 3-step SARSA with fast and slow sub-networks. Specify the corresponding online update rule using **full gradient**.

Loss L' = $\frac{1}{2}\delta^2$ with $\delta = [r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 Q_{\text{slow}}(\mathbf{x}_{t+3}, a_{t+3}) - Q_{\text{fast}}(\mathbf{x}_t, a_t)]$

Q_{fast} update: $\Delta w_{ij} = \eta\delta \frac{\partial Q_{\text{fast}}}{\partial w_{ij}}(\mathbf{x}_t, a_t)$

Q_{slow} update: $Q_{\text{slow}} \leftarrow Q_{\text{fast}}$ every C time steps

number of points: ...../ 1.5

(vi) What would be different in your answer to (v) if you used **semi-gradient** instead of full gradient? Give a reason.
The answer does not change. Full-gradient and semi-gradient update are equivalent in the case since we use fast and slow sub-networks, i.e. the target $Q(\mathbf{x}_{t+3}, a_{t+3}) = Q_{\text{slow}}(\mathbf{x}_{t+3}, a_{t+3})$ is considered fixed during the regular gradient updates.

number of points: ...../ 0.5

(vii) Is 3-step deep SARSA an on-policy or an off-policy algorithm?
On-policy

number of points: ...../ 0.5

(viii) What changes if you want to write an analogous loss function to (i) for a deep network implementing 3-step Q-learning?
We have to replace $Q_{\mathbf{w}}(s_{t+3}, a_{t+3})$ by $\max_{a'} Q_{\mathbf{w}}(s_{t+3}, a')$ in the loss of (i).

number of points: ...../ 1

(ix) Your friend claims that at least one of the two algorithms (3-step SARSA, 3-step Q-learning) converges "in expectation". Among the two, pick an algorithm that converges in expectation. Using one of the two proof techniques that you have seen in class, show that, when the algorithm has converged, its Q-values are Bellman-consistent.
3-step SARSA converges in expectation, i.e.

$$\begin{aligned} 0 &= E[\Delta Q(s,a)|s,a] \\ &= E[r(s,a) + \gamma r(s',a') + \gamma^2 r(s'',a'') + \gamma^3 Q(s''',a''') - Q(s,a)|s,a] \\ &= E[r(s,a) + \gamma r(s',a') + \gamma^2 r(s'',a'') + \gamma^3 Q(s''',a''')] - Q(s,a), \end{aligned}$$

such that

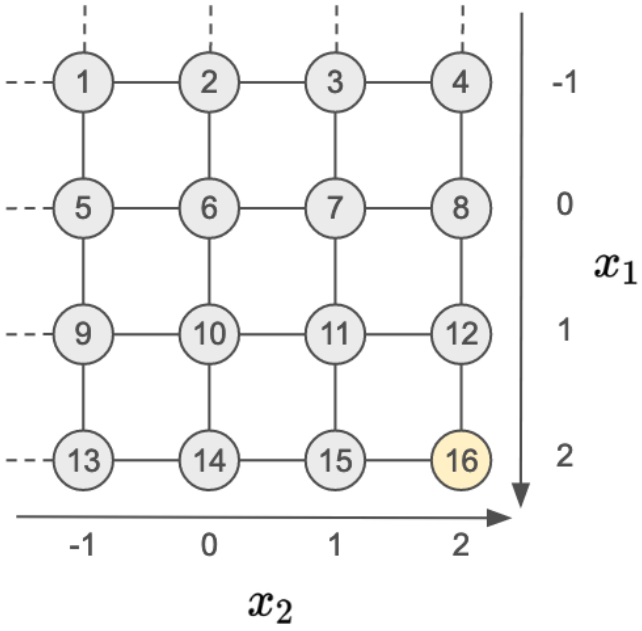
$$\begin{aligned} Q(s,a) &= \sum_{s'} P^a_{s\to s'} \left(R^a_{s\to s'} + \gamma E[\gamma r(s',a') + \gamma^2 r(s'',a'') + \gamma^3 Q(s''',a''')|s'] \right) \\ &= \sum_{s'} P^a_{s\to s'} \left(R^a_{s\to s'} + \gamma \sum_{a'} \pi(a'|s') \sum_{s''} P^{a'}_{s'\to s''} \left(R^{a'}_{s'\to s''} + E[\gamma r(s'',a'') + \gamma^2 Q(s''',a''')|s''] \right) \right) \\ &= \sum_{s'} P^a_{s\to s'} \left(R^a_{s\to s'} + \gamma \sum_{a'} \pi(a'|s') \sum_{s''} P^{a'}_{s'\to s''} \left(R^{a'}_{s'\to s''} + \gamma \sum_{a''} \pi(a''|s'') \sum_{s'''} P^{a''}_{s''\to s'''} \right.\right. \\ &\qquad \left.\left. \left(\sum_{s'''} R^{a''}_{s''\to s'''} + \gamma \sum_{a'''} \pi(a'''|s''') Q(s''',a''') \right) \right) \right). \end{aligned}$$

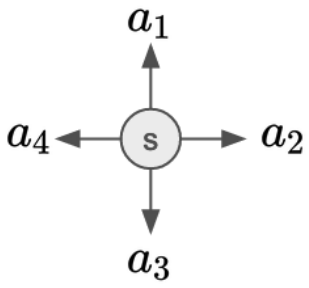
The last line is equivalent to the 3-step Bellman equation, which gives the Bellman consistency of the algorithm.

3-step Q-learning is only Bellman-consistent under the assumption of an ϵ-greedy policy (consistency proof analogous to above).

number of points: ...../ 3.5

4 Inference Prior and Reinforcement Learning (10 points)





We consider a 2-dimensional discrete environment (Figure). In the figure, the bottom right corner of an infinite grid is shown (the grid is infinite in the negative direction of x_1 and x_2). State 16 is the goal state, where the agent receives a positive reward r. All other states give a reward of zero. Available actions (figure on the right) are a1=up, a2=right, a3=down, a4=left (whenever these moves are possible).

Suppose that we use function approximation for $Q(a, X) = \sum_j w_{a,j} x_j$ with continuous state representation $X = (x_1, x_2, x_3)$ where x_1 and x_2 are the coordinates of the state in the grid as shown in the figure, and $x_3 = 1$ for all states. For instance, the state 15 in the figure is encoded as $(2, 1, 1)$.

Before the first episode, we initialize all weights at zero. During the first episode, we update Q-values using the online Q-learning algorithm in continuous space derived with the semi-gradient method from the Q-learning error function. We use a learning rate η such that $0 < \eta < 0.2$.

(i) Write down the quadratic loss function for 1-step Q-learning.

$\frac{1}{2}\delta^2$ with $\delta = (r + \gamma \max'_a Q(X, a') - Q(X, a))$

number of points: ...../ 0.5

(ii) In the first episode, the agent starts in state 11, takes action a_2 to state 12, and then takes action a_3 and arrives at the goal state, obtaining a reward r and finishing the episode. Using the semi-gradient update rule, what are the new weight values w_{ai} at the end of the episode? **Write down all weights that have changed.**

$w_{a_3,1} = \eta r$, $w_{a_3,2} = 2\eta r$, $w_{a_3,3} = \eta r$

number of points: ...../ 2

(iii) In the second episode, the agent starts again in state 11 and uses a greedy policy. In case of ties, the agent will select action a_2. Using the semi-gradient update rule, what are the new weight values w_{ai} at the end of the second episode? **Write down all weights that are non-zero after the second episode.**

$w_{a_2,1} = 2\eta r$, $w_{a_2,2} = \eta r$, $w_{a_2,3} = \eta r$
$w_{a_3,1} = \eta r - 4\eta^2 r$, $w_{a_3,2} = 2\eta r - 4\eta^2 r$, $w_{a_3,3} = \eta r - 4\eta^2 r$

number of points: ...../ 2

(iv) After two episodes, the agent is considered to be fully trained and **stops updating the weights** with new experiences. In the third episode, the agent starts at state 6 and follows a greedy strategy. What will be the episode trajectory? Write all transitions of the episode (state, action, state, action, ...).

Two trajectories are possible (unique trajectory only exists for $0 < \eta < 0.125$):
(6, a2, 7, a2, 8, a3, 12, a3, 16)
(6, a2, 7, a3, 11, a2, 12, a3, 16)

number of points: ...../ 1.5

(v) What can you say about the inference prior of the variables x_1, x_2 and x_3?

In simple words, the inference prior of variable x_1 is

that the good action is similar among all states with positive x_1 coordinate, but different from states with negative x_1.

In simple words, the inference prior of variable x_2 is

that the good action is similar among all states with positive x_2 coordinate, but different from states with negative x_2.

In simple words, the inference prior of variable x_3 is

that the good action is the same for all states (and it is balancing x_1 and x_2, similar to an optimistic initialization).

number of points: ...../ 2

(vi) Suppose now that the agent starts a fourth episode in state 1, and follows a greedy strategy. In case of ties, the agent selects a random action among the best actions. What would the episode look like? Relate this to your answer in (v).

The agent will select randomly between a_1 and a_4 at each step, so the agent will go in the negative direction forever.

number of points: ...../ 1

(vii) How could we modify x_3 to change the behavior in (vi)? What would change?

We could increase x_3, so that actions a_2 and a_3 would still be considered good in state 1. In this case, the agent would reach the goal state.

number of points: ...../ 1

Free space for your calculations, do not use to write down answers.