

Last Name

First Name.....

Artificial Neural Networks / Reinforcement Learning: Exam 17th of June 2024

- 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

This page remains empty. You can use it as free space for your calculations, do not use to write down answers.

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

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

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:

Update 2:

Update 3:

Update 4:

Update 5:

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)

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)

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)

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:

$$E[r(s, a)|\mathbf{w}] = \dots\dots\dots$$

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.

$$\nabla_{\mathbf{w}}E[r(s, a)|\mathbf{w}] = \dots\dots\dots$$

number of points:/ 2

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

$$\Delta w_{nm} = \dots\dots\dots$$

number of points:/ 1

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

.....

number of points:/ 1

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

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

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} :

$$\Delta w_{nm} = \dots\dots\dots$$

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 It can be interpreted as

The 2nd factor is It can be interpreted as

The 3rd factor is It can be interpreted as

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 $L_{bt} = \dots\dots\dots$

Online: Loss $L_{on} = \dots\dots\dots$

number of points:/ 1

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

.....

number of points:/ 0.5

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

Update:

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?

.....

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 $L' = \dots\dots\dots$

Q_{fast} update:

Q_{slow} update:

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.

.....
.....

number of points:/ 0.5

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

.....

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?

.....

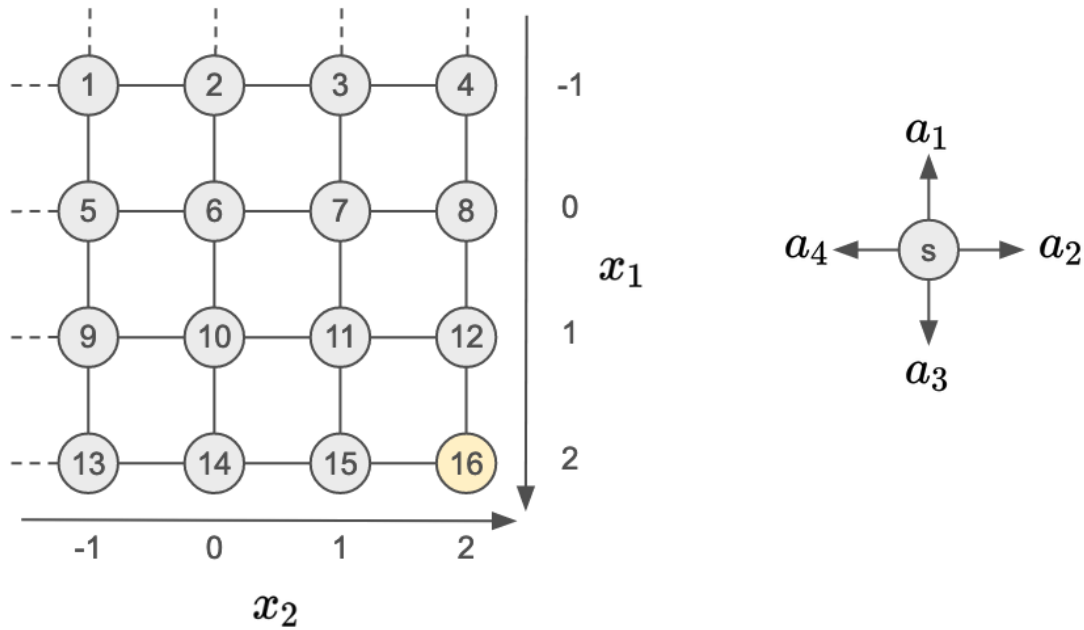
number of points:/ 1

(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.

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

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 a_1 =up, a_2 =right, a_3 =down, a_4 =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.

.....

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.**

.....
.....
.....

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.**

.....
.....
.....

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, ...).

.....
.....

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

.....

In simple words, the inference prior of variable x_2 is

.....

In simple words, the inference prior of variable x_3 is

.....

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).

.....
.....

number of points:/ 1

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

.....

number of points:/ 1

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

This page is empty. Free space for your calculations, do not use to write down answers.