Exam CS4400: Deep Reinforcement Learning

21-06-2022 | 9:00-11:30

Student name: ____

Student number: ____________________

- This is a closed-book individual examination with **10 questions** and a total of **50 points**.
- Do not open the exam before the official start of the examination.
- If you feel sick or otherwise unable to take the examination, please indicate this *before* the exam starts.
- The examination lasts **150 minutes** after the official start.
- This gives you roughly 3 minutes per point. Use your time carefully!
- You can hand in your exam solution any time until 15 minutes before the end of the exam and leave the examination room quietly. In the last 15 minutes, no one can leave the examination room to help other students concentrate on finishing their exam.
- Only one student can visit the bathroom at the same time. In the last 15 minutes, no bathroom visits are possible.
- Use of course books, readers, notes, and slides is **not** permitted
- Use of (graphical) calculators or mobile computing devices (including mobile phones) is **not** permitted.
- Write down your name and student number above.
- Write your **student number on each sheet** of the exam after the exam started.
- You can write your answer on the free space under each question.
- If you need more space, use the back of another exam-sheet and write where to find the answer under the question. Ask for additional empty pages if you need them.
- Use pens with black or blue ink. Pencils and red ink are not allowed!
- Clearly cross out invalid answers. If two answers are given, we consider the one with less points!
- Write clearly, use correct English, and avoid verbose explanations. Giving irrelevant information may lead to a reduction in your score.
- This exam covers all information on the slides of the course, the tutorials and everything discussed in lectures.
- This exam assumes a familiarity with the stated background knowledge of the course.
- The total number of pages of this exam is 11 (excluding this front page).
- Exam prepared by Wendelin Böhmer.

Question 1 (multiple choice): **(18 points)**

Please mark only the correct answers with a **cross** like this: ☒. If you wish to **unmark** a marked answer, **fill** the entire square and **draw an empty** one next to it like this: ☐ ■

Only one answer per question is correct. You will receive 1 point per correct answer, except if multiple squares are marked. Wrong answers yield no points, but are also not punished. Good luck!

1.1: Which of the following definitions is called the *mean squared loss* over samples x and labels y?

☐ $-\mathbb{E}[\ln p(y|x) \,|\, (x,y) \sim \mathcal{D}]$
☐ $-\mathbb{E}[p(y|x)\; \ln p(y|x) \,|\, (x,y) \sim \mathcal{D}]$
☐ $\mathbb{E}[(f(x)-y)^2 \,|\, (x,y) \sim \mathcal{D}]$
☐ $\sqrt{\mathbb{E}[(f(x)-y)^2 \,|\, (x,y) \sim \mathcal{D}]}$

1.2: Which of the following is **not** a parameter of the pytorch `MaxPooling2d` constructor?

☐ `kernel_size`
☐ `padding`
☐ `stride`
☐ `out_channels`

1.3: How large is the output of a convolutional layer with a $4 \times 3 \times 5 \times 5$ kernel when applied to a 30×20 RGB image?

☐ $3 \times 26 \times 16$
☐ $4 \times 26 \times 16$
☐ $3 \times 34 \times 24$
☐ $4 \times 34 \times 24$

1.4: For a linear function $f(\boldsymbol{x}) = \boldsymbol{a}^\top \boldsymbol{x} + b$, which of the following will improve generalization?

☐ larger $\|\boldsymbol{a}\|$
☐ smaller $\|\boldsymbol{a}\|$
☐ larger b
☐ smaller b

1.5: Which of the following is the *off-policy* state-action value target for policy π?

☐ $r_t + \gamma \max_a Q(s_{t+1}, a)$
☐ $r_t + \gamma Q(s_{t+1}, \arg\max_a Q'(s_{t+1}, a))$
☐ $r_t + \gamma Q(s_{t+1}, a_{t+1})$
☐ $r_t + \gamma \mathbb{E}[Q(s_{t+1}, a) \,|\, a \sim \pi(\cdot|s_{t+1})]$

1.6: Which of the following is called the *maximum entropy policy* of Q-value function $Q^\pi(s,a)$?

- ☐ $\pi'(a|s) = (1-\epsilon)\,\delta\big(a = \arg\max_{a'} Q^\pi(s,a)\big) + \epsilon\frac{1}{|\mathcal{A}|}$
- ☐ $\pi'(a|s) = \frac{\exp(\epsilon Q^\pi(s,a))}{\sum_{a'}\exp(\epsilon Q^\pi(s,a'))}$
- ☐ $\pi'(a|s) = \exp\left(\frac{1}{\epsilon}Q^\pi(s,a) - \frac{1}{\epsilon}V^\pi(s)\right)$
- ☐ $\pi'(a|s) = \arg\max_\pi Q^\pi(s,a) \quad \text{s.t.} \quad D_{\text{KL}}[\mu(\cdot|s)\|\pi(\cdot|s)] \leq \delta$

1.7: Which algorithm does **not** use an experience replay buffer?

- ☐ Deep deterministic policy gradients (DDPG)
- ☐ Soft actor-critic (SAC)
- ☐ Proximal policy optimization (PPO)
- ☐ Deep Q-networks (DQN)

1.8: Which of the following is a condition for local convergence of actor-critic methods?

- ☐ the learning rate of the critic must be higher than that of the actor
- ☐ the critic must be trained with a low variance estimator like TD(0)
- ☐ the critic of linear policies must be non-linear
- ☐ the action space must be continuous

1.9: Which of the following algorithms or techniques uses the *reparameterization trick*?

- ☐ Trust-region policy optimization (TRPO)
- ☐ Twin delayed DDPG (TD3)
- ☐ Upper confidence bounds (UCB)
- ☐ Bayes by backpropagation

1.10: Which exploration method is best suited to learn a multi-armed bandit?

- ☐ ϵ-greedy
- ☐ Boltzmann policy
- ☐ optimistic exploration
- ☐ deep exploration

1.11: Which of the following statements about deep exploration is correct?

- ☐ deep exploration learns faster than undirected exploration
- ☐ intrinsic reward is a form of optimistic exploration
- ☐ deep exploration always explores all states before converging to the optimal policy
- ☐ Thompson sampling can not perform deep exploration

1.12: Which of the following uncertainty estimates can be used for Thompson sampling?

☐ pseudo-counts
☐ random network distillation
☐ random hash functions
☐ ensembles

1.13: What is the *behavioural cloning* (offline) error bound for

$$\epsilon = \mathbb{E}_{\mathcal{D}}\big[\delta(a_t = a_t^*)\big|_{a_t \sim \pi_\theta(\cdot|s_t)}\big] \quad \text{and} \quad f(H,\theta) := \mathbb{E}_{\pi_\theta}\Big[\sum_{t=0}^{H-1} \delta(a_t = a_t^*)\Big|_{\substack{a_t \sim \pi_\theta(\cdot|s_t) \\ a_t^* \sim \pi^*(s_t)}}\Big] \,?$$

☐ $f(H,\theta) \leq C + H\epsilon$
☐ $f(H,\theta) \leq C + H^2\epsilon$
☐ $f(H,\theta) \leq C + H\epsilon^2$
☐ $f(H,\theta) \leq C + H^2\epsilon^2$

1.14: Which of the following is a possible value target $\underline{Q}(s,a)$ for *pessimistic offline* DQN with an ensemble of Q-value functions Q_{θ_i}?

☐ $\underline{Q}(s,a) := Q_{\theta_1}(s, \arg\min_{a'} Q_{\theta_2}(s,a'))$
☐ $\underline{Q}(s,a) := Q_{\theta_1}(s, \arg\max_{a'} Q_{\theta_2}(s,a'))$
☐ $\underline{Q}(s,a) := \mathbb{E}[Q_{\theta_i}(s,a)] + \alpha\sqrt{\mathbb{V}[Q_{\theta_i}(s,a)]}$
☐ $\underline{Q}(s,a) := \mathbb{E}[Q_{\theta_i}(s,a)] - \alpha\sqrt{\mathbb{V}[Q_{\theta_i}(s,a)]}$

1.15: Which property can lead to *cyclic games*?

☐ discrete state spaces
☐ continuous state spaces
☐ equal information
☐ unequal information

1.16: Which effect can lead to problems when two autonomously driving cars try to cross an intersection? You can assume that the cars are from the same manufacturer and have been trained using IQL with strong punishment for crashes.

☐ centralized training
☐ zero-shot coordination
☐ relative overgeneralization
☐ value factorization

1.17: In a Dec-POMDP with 2 agents and $|\mathcal{A}^i| = 3$, $\forall i$, how many heads are on a *joint* Q-value function with a DQN architecture?

☐ 1
☐ 3
☐ 8
☐ 9

1.18: Which of the following algorithms does **not** use *value factorization*?

- ☐ MADDPG
- ☐ VDN
- ☐ QMIX
- ☐ DCG

Question 2:

(3 points)

Explain in no more that 5 sentences at least **three** differences between TD3 and SAC.

Question 3: **(2 points)**

Describe in 4 sentences or less at least **two** different things that can hurt learning when exploring with ϵ-greedy.

Question 4: **(2 points)**

Define in 4 sentences or less the causes of *aleatoric* and *epistemic* uncertainty, and the difference between between.

Question 5:

(2 points)

Using 4 or less sentences, name and explain **two** distinct advantages of the *centralized training decentralized execution* paradigm for cooperative MARL.

Question 6:

(3 points)

Give the joint actions of all Nash-equilibria for a two-player single-state general-sum game with the following reward matrix, where entry x/y denotes the reward x for player 1 and y for player 2:

P1/P2	a_1^2	a_2^2	a_3^2	a_4^2
a_1^1	4/2	5/8	1/3	2/7
a_2^1	2/4	7/5	1/4	1/4
a_3^1	7/2	4/6	7/1	3/2
a_4^1	2/3	6/4	3/1	4/5

Make sure you provide the *joint actions* of all Nash equilibria, not just the obtained rewards. Which Nash equilibrium would player 1 prefer? Which would be better for player 2?

Question 7: **(4 points)**

Let $f : \mathbb{R} \to \mathbb{R}$ denote a random function, where the output $f(x) \sim \mathcal{N}(\mu(x), \sigma^2)$ is drawn i.i.d. for each input $x \in \mathbb{R}$. Prove analytically that the expected *mean squared error* $\mathbb{E}[\mathcal{L}^{\text{mse}}]$ for a given data-set $\{x_i, y_i\}_{i=1}^n$ is:

$$\mathbb{E}[\mathcal{L}^{\text{mse}}] \quad = \quad \frac{1}{n}\sum_{i=1}^{n}\big(\mu(x_i) - y_i\big)^2 + \sigma^2$$

Note that for each index i, the data tuple $\langle x_i, y_i \rangle$ is fixed, whereas the function output $f(x_i)$ is random!

Question 8: **(5 points)**

Let $\mathcal{G} := \langle \mathcal{V}, \mathcal{E} \rangle$ denote a graph with n nodes and edges $(v, v') \in \mathcal{E}$ from nodes $v \in \mathcal{V}$ to nodes $v' \in \mathcal{V}$. Let furthermore $\mathbf{V} \in \mathbb{R}^{n \times d}$ denote the d-dimensional node-annotations collected into one matrix and $g(\mathbf{V}, \mathcal{E}, \mathbf{B}) \in \mathbb{R}^{n \times b}$ denote a *graph convolutional layer*, with given message parameters $\mathbf{B} \in \mathbb{R}^{d \times b}$, defined as

$$g(\mathbf{V}, \mathcal{E}, \mathbf{B})_{m,k} \;:=\; \sum_{l=1}^{n} \sum_{p=1}^{d} W_{m,l}\, V_{l,p}\, B_{p,k}\,, \qquad 1 \leq m \leq n\,, \qquad 1 \leq k \leq b\,.$$

Define topology matrix $\mathbf{W} \in \mathbb{R}^{n \times n}$ and the *linear function* $f : \mathbb{R}^{\mathcal{J}} \to \mathbb{R}^{\mathcal{I}}$ equivalent to $g(\mathbf{V}, \mathcal{E}, \mathbf{B})$:

$$f(\boldsymbol{z})_i \;:=\; \sum_{j \in \mathcal{J}} \Theta_{i,j}\, z_j\,, \qquad \forall \boldsymbol{z} \in \mathbb{R}^{\mathcal{J}}\,, \quad \forall i \in \mathcal{I}\,.$$

f should be defined by constructing the index sets $\mathcal{J}$ and $\mathcal{I}$, the inputs $\boldsymbol{z}$ from $\mathbf{V}$, and the parameter matrix/tensor $\boldsymbol{\Theta} \in \mathbb{R}^{\mathcal{I} \times \mathcal{J}}$ from g's parameters $\mathbf{B}$ and topology matrix $\mathbf{W}$.

Hint: if you do not know how to define $\mathbf{W}$, just write that you assume $\mathbf{W}$ is known and continue.

Question 9: **(4 points)**

Given a POMDP $M := \langle \mathcal{S}, \mathcal{A}, \mathcal{O}, \rho, P, R, O \rangle$, and a belief distribution $b(s|\tau_t)$, with τ_t denoting the observation-action history at time t, define the MDP $M' := \langle \mathcal{S}', \mathcal{A}', \rho', P', r' \rangle$ over the sufficient statistics of the belief b, where r' is the average reward function of M' (the full reward distribution R' can be omitted). Make sure you define *all* the components of M'.

Hint: the MDP M' is defined in terms of observation-action histories, as in DRQN.

Question 10: (programming) **(7 points)**

You only have to insert the missing code segment at the line(s) marked with *#YOUR CODE HERE*. Please use correct Python/PyTorch code. Singleton dimensions of tensors can be ignored, i.e., you do not need to (un)squeeze tensors. If you forget a specific command, you can define it first, both the signature (input/output parameters) and a short description what it does. Using your own definitions of existing PyTorch functions will not yield point deductions. If no similar PyTorch function exists, your definition will be considered as wrong code and you will not receive the corresponding points.

Implement the following *value loss* for the DDPG algorithm in the given `MyLearner` class **efficiently**:

$$\min_{\phi} \quad \mathbb{E}\Big[\frac{1}{\sum_{i=1}^{m} n_i}\sum_{i=1}^{m}\sum_{t=0}^{n_i-1}\Big(r_t^i + \gamma Q_{\phi'}\big(s_{t+1}^i, \boldsymbol{\pi}_\theta(s_{t+1}^i)\big) - Q_\phi(s_t^i, a_t^i)\Big)^2 \,\Big|\, \tau_{n_i}^i \sim \mathcal{D}\Big],$$

where $\tau_{n_i}^i := \{s_t^i, a_t^i, r_t^i\}_{t=0}^{n_i-1} \cup \{s_{n_i}^i\}$, denotes m trajectories of states $s_t^i \in \mathbb{R}^d$, actions $a_t^i \in \mathbb{R}^b$ and rewards $r_t^i \in \mathbb{R}$. The last state $s_{n_i}^i$ is always terminal. The target network parameters shall be $\phi' := \phi$ at all times, but no gradients shall flow into ϕ'!

Hint: Efficient implementations produce minimal computation graphs. You can use the given function `self.values()` to compute the Q-values $Q_\phi(s, a)$ for a mini-batch (tensor) of states s_t^i and a mini-batch (tensor) of the corresponding actions a_t^i, with the same size (except for the last dimension). The deterministic policy $\boldsymbol{\pi}_\theta$ is computed by another given module `policy` that takes a mini-batch (tensor) of states s_t^i as input and outputs an equally sized (except in the last dimension) tensor of actions $a_t'^i = \boldsymbol{\pi}_\theta(s_t^i)$.

```python
import torch

class MyLearner:
    def __init__(self, model, policy, gamma=0.99):
        self.value_model = model
        self.policy_model = policy
        self.gamma = gamma
        self.all_parameters = [p for m in self.value_models for p in m.parameters()]
        self.optimizer = torch.optim.Adam(self.all_parameters)

    def values(self, states, actions):
        return self.value_model(torch.cat([states, actions], dim=-1))

    def train(self, batch):
        """ Performs one gradient update step on the loss defined above.
            "batch" is a dictionary of equally sized tensors
            (except for last dimension):
                - batch['states'][i, t, :] = s_t^i
                - batch['actions'][i, t, :] = a_t^i
                - batch['rewards'][i, t] = r_t^i
                - batch['mask'][i, t] = t < n_i
                - batch['terminals'][i, t] = s_t^i is terminal """
        loss = 0
        # YOUR CODE HERE
        self.optimizer.zero_grad()
        loss.backward()
        self.optimizer.step()
        return loss.item()
```

You can use the next page to write down your answer as well!

Question 10 (continuation):

End of exam. **Total 50 points.**