RL exam

Spring 2024 RL Exam: Key Questions

Here are the questions I remember from the Spring 2024 RL Exam:

1. **What is the deadly triad in RL?** Explain each component.
2. Write down the Bellman Expectation and Bellman Optimality Equations for the Q-value function.
3. Rank RL algorithms by sample efficiency. Choices: Policy Gradients, MBRL, DQN.
4. Write down the Vanilla Policy Gradient formula.
5. Write the batch-constrained policy action formula.
6. Explain two reasons why intrinsic exploration is necessary in RL.
7. Why can't we directly apply off-policy algorithms to offline RL problems?
8. Can imitation learning be considered a special case of offline RL?
9. How do PPO and TRPO improve upon vanilla policy gradients?
10. What is the "Noisy-TV" problem in exploration?
11. What networks are used in AlphaGo's deployment?
12. Which is better: every-visit MC or first-visit MC?

Programming Questions:

1. Identify 3 mistakes in the Double DQN implementation, primarily related to using the correct Q-value function in updates.
2. Implement Thompson Sampling from scratch.

Exam Tips:

It's crucial to thoroughly review the course slides and complete every programming assignment. However, even doing this might not guarantee a perfect 1.0 grade. Some questions require prior RL knowledge beyond the course material. Also, be ready to recall and write down key formulas (e.g., Bellman, Policy Gradient) during the exam.

Sample Exams

Unfortunately, past exam papers aren't provided. However, you can find RL exams from other universities online. While the content may differ, practicing these can be beneficial. Here are some samples:

- Midterm 2 Solution - Spring 2020
- Final Exam - Spring 2020

Recommended Online Courses

Several online RL courses are worth checking out:

1. **Stanford CS234**: This course is closely related to the one at FAU. For a more advanced level, consider CS285 by Sergey Levine.
2. **Foundations of Deep RL** by Pieter Abbeel: The first four lectures are especially relevant.
3. **Introduction to RL** by Benjamin Eysenbach: The written notes are insightful, and the exercises are definitely worth doing.
4. **DeepMind x UCL RL Lecture Series**: Great explanations of algorithms like value iteration and UCB, which are intuitive and easy to follow.

Q&A Section

Q1: The state space is two-dimensional, and one dimension significantly affects the value function more than the other. How would you design the tilings to leverage this prior knowledge?
A1: To emphasize generalization across x_1, use finer tilings along x_1 and coarser tilings along x_2. This means each tile will cover a smaller range in x_1 and a larger range in x_2.

Q2: What does the Advantage function measure?
A2: The Advantage Function measures how much better or worse taking a particular action a in state s is compared to the average action in that state. $V(s)$ gives the expected value of the state, while $Q(s, a)$ gives the value for taking that action in the state.
A(s,a)=Q(s,a)−V(s)A(s,a) = Q(s,a) - V(s)A(s,a)=Q(s,a)−V(s)

Q3: Which helps our agent more:
a) Information about *invariant actions* in *good states*, or
b) Information about *good actions* in *challenging states*?
A3: a) is more informative as it guides exploration and enforces the distinction between good and bad actions.

Q4: What are the sources of distribution mismatch in offline RL?
A4:

1. The learned policy creates a different state-visitation distribution compared to the behavior distribution.
2. During policy improvement, the learned policy requires Q-function evaluation on unseen actions (out-of-distribution). Overestimation of Q-values on these samples leads to learning a suboptimal policy (extrapolation error).

Q5: What is the main idea behind DeepHashing?
A5: Instead of counting high-dimensional states (images), we count the latent compressed

states. Map a state to a hash code, then count up states visited with that hash code. Encourage visiting states with low-count hash codes.

Q6: What is the limitation of prediction error as a bonus in ICM (Curiosity-driven Exploration by self-supervised prediction)?
A6: Even with an ICM module (using inverse dynamics instead of autoencoders), the agent is rewarded indefinitely for noise states with unpredictable outcomes, leading to attraction to "Noisy TV" situations.

Q7: What is the main idea behind Go-Explore?
A7:

- **Detachment:** An agent driven by intrinsic motivation (IM) could detach from high-IM regions and forget how to return (catastrophic forgetting). Go-Explore solves this by explicitly storing an archive of promising states so they can be revisited later.
- **Derailment:** When an agent discovers a promising state, it should return to it and explore from there. However, IM may prevent agents from returning to these states.

Q8: How can we reduce variance in Policy Gradients?
A8:

- **Causality (reward-to-go):** Only penalize actions after time step t.
- **Generalized Advantage Function:** Use the advantage function in the policy gradient equation.

Q9: What is bootstrapping in RL?
A9: Bootstrapping refers to using Q-values as targets when updating Q-values.

Q10: Why is Q-learning off-policy?
A10: Q-learning is off-policy because the target value it computes for updates doesn't depend on the current policy. Instead, it uses the maximization operator to choose the action at s'. This contrasts with policy gradient methods, which require $\log \pi_\theta$ at the current θ.

Q11: What are the problems in classic Q-learning?
A11: Q-learning suffers from maximization bias. Double Q-learning addresses this by using one Q-value for maximization and another for evaluation: r+γQ1(s′,argmaxaQ2(s′,a))r + \gamma Q_1(s', \text{argmax}_a Q_2(s', a))r+γQ1(s′,argmaxaQ2(s′,a)) This breaks the maximization bias.

Q12: What is the deadly triad in deep RL?
A12:

- **Function Approximation**
- **Bootstrapping:** Using model outputs as regression targets.
- **Off-Policy Learning:** Updating using transitions not generated by the policy.

Tabular Q-learning already incorporates bootstrapping and off-policy learning. DQN adds function approximation.

Q13: How does DQN counter the deadly triad?
A13:

1. **Replay Buffer:** Minimizes the impact of temporal correlations and allows multiple passes through the same data points.
2. **Target Network:** This network is updated infrequently to stabilize the target, mitigating the problem of constantly changing targets.