Homework #3
(Value Function Approximation)

INF8250AE – Introduction to Reinforcement Learning
(Fall 2025)

- **Deadline:** Monday, December 1, 2025 at **16:59**.
- **Submission:** You need to submit two files through Gradescope. One is a PDF file including all your answers and plots. The other is a source file that reproduces your answers. You can produce the file however you like (e.g. LaTeX, Microsoft Word, etc) as long as it is readable. Points will be deducted if we have a hard time reading your solutions or understanding the structure of your code. If the code does not run, you may lose most/all of your points for that question.
- **Late Submission:** 10% of the marks will be deducted for each day late, up to a maximum of 3 days. After that, no submissions will be accepted.
- **Collaboration:** Homework assignments can be done in a group of at most two students. You must clearly specify the role of each team member in solving the assignment. You should not use LLMs to answer the questions.

1 A Random Dose of Theory [20pt]

Exercise 1. [pt10] TD Learning from Finite MDPs to Linear Function Approximator and Back

Recall that the TD update rule for a linear value function approximator of the form $V(x; w) = \phi^\top(x)w$ *after observing data* (X_t, R_t, X_{t+1}) *is*

$$w_{t+1} \leftarrow w_t + \alpha\delta_t\phi(X_t),$$

with $\delta_t = R_t + \gamma\phi(X_{t+1})^\top w_t - \phi(X_t)^\top w_t$ *being the TD error (here we do not specify the time-dependence of the learning rate* α*). Let us compare it with the TD Learning for finite MDPs, which is*

$$V(X_t) \leftarrow V(X_t) + \alpha[R_t + \gamma V(X_{t+1}) - V(X_t)].$$

1. **[4pt]** *What are the similarities and differences between these two update rules?*

2. **[4pt]** *Consider a finite MDP with* $N = |\mathcal{X}|$*. Choose* ϕ *such that any value function of the finite MDP can be represented in the form of* $V(x; w) = \phi^\top(x)w$*. Specify the dimensions of* ϕ *and* w*.*

3. **[2pt]** *What is the update rule for TD with linear VFA under this particular choice of* ϕ*? Is it the same or different from the TD learning for finite MDPs?*

Exercise 2. [pt10] The Projected Life of the TD Solution

Let π *be a fixed policy with stationary distribution* ρ^π *and* $\mathcal{F}$ *be a linear function class. Recall that the operator* $\Pi_{\mathcal{F},\rho^\pi}$ *denotes the orthogonal projection onto* $\mathcal{F}$ *with respect to the inner product*

$$\langle f, g \rangle_{\rho^\pi} = \sum_x \rho^\pi(x) f(x) g(x).$$

The fixed point of the projected Bellman operator,

$$V_{\mathrm{TD}} \;=\; \Pi_{\mathcal{F},\rho^\pi} T^\pi V_{\mathrm{TD}},$$

is the value function returned by LSTD or linear TD(0).

We can upper bound the difference between V_{TD} *and the true value function* V^π *as a function of how well* V^π *can be represented within* $\mathcal{F}$ *as*

$$\|V_{\mathrm{TD}} - V^\pi\|_{2,\rho^\pi} \leq \frac{\|\Pi_{\mathcal{F},\rho^\pi} V^\pi - V^\pi\|_{2,\rho^\pi}}{\sqrt{1-\gamma^2}}.$$

This result states that V_{TD} *is within a factor* $1/\sqrt{1-\gamma^2}$ *of the best approximation to* V^π *available in* $\mathcal{F}$*.*

1. **[2pt]** *Explain why* $\|\Pi_{\mathcal{F},\rho^\pi} V^\pi - V^\pi\|_{2,\rho^\pi}$ *shows "how well* V^π *can be represented within* $\mathcal{F}$*", as stated above.*

2. **[8pt]** *Prove this result.*

Hints:

1. *Start from* $V_{\mathrm{TD}} - V^\pi$*, and add and subtract the projection of* V^π *onto* $\mathcal{F}$*.*

2. *Benefit from the fact that* $V_{TD} = \Pi_{\mathcal{F},\rho^\pi} T^\pi V_{TD}$*.*

3. *Take the* $L_2(\rho^\pi)$*-norm squared of both sides and use*

$$\|a + b\|^2_{2,\rho^\pi} = \|a\|^2_{2,\rho^\pi} + \|b\|^2_{2,\rho^\pi} + 2 \langle a , b \rangle_{2,\rho^\pi} ,$$

   with an appropriate choice of a *and* b*.*

4. *Show that the inner product term is zero. Benefit from the fact that* $\Pi_{\mathcal{F},\rho^\pi} V^\pi - V^\pi$ *is orthogonal to* $\mathcal{F}$*.*

5. *Show that projected Bellman difference can be bounded as*

$$\|\Pi_{\mathcal{F},\rho^\pi} T^\pi V_{\mathrm{TD}} - \Pi_{\mathcal{F},\rho^\pi} V^\pi\|_{2,\rho^\pi} \leq \gamma \|V_{\mathrm{TD}} - V^\pi\|_{2,\rho^\pi} .$$

   Whenever you use a non-trivial result, you need to refer to the exact source of it in the Foundations of Reinforcement Learning book (lemma or theorem name + section number); otherwise, you may lose some marks.

2 Lest We Forget DQN [80pt + 5(Bonus)pt]

In this part of the homework, you will implement a simplified version of the Deep Q-Network by Mnih et al. [2015], study some of its pitfalls, and explore some variations of it. To limit the running time, we will learn directly from the state representations rather than visual observations, as opposed to what Mnih et al. [2015] do. All other elements, however, will be the same. We will be learning a Q_θ function with parameters θ using stochastic gradient descent (SGD) and nonlinear function approximation, specifically, a deep neural network (DNN). DQN is a practical implementation of the Fitted Q-Iteration (FQI)/Approximate Value Iteration algorithm (AVI) covered in lecture, but with a couple of modifications to make it suitable for a DNN.

We will implement a replay buffer and a target network. A replay buffer, $\mathcal{D}$, is used to store transitions observed while interacting with the environment. The replay buffer is sampled during training to update the Q function.

The target network, $Q_{\theta'}$ with parameters θ' is used to compute the target values in our loss. In DQN, the target network is held fixed for k number of steps and then updated to have the same weights as Q_θ by copying $\theta' \leftarrow \theta$.

The loss function for training the value function is

$$L(\theta) = \mathbb{E}_{(X_t, A_t, R_t, X_{t+1}) \sim \mathcal{D}} \left[\left| R_t + \gamma \max_{a' \in \mathcal{A}} Q_{\theta'}(X_{t+1}, a') - Q_\theta(X_t, A_t) \right|^2 \right],$$

with $(X_t, A_t, R_t, X_{t+1}) \sim \mathcal{D}$ meaning that the tuple is samples from the replay buffer $\mathcal{D}$.

Exercise 3. [20pt] A Dose of Q-riosity: Exploring DQN through Short Questions

Answer these short questions.

1. **[4pt]** *What are the similarities and differences between DQN and the AVI/FQI framework introduced in the class? And what about its similarities and differences with Q-Learning? Briefly discuss.*

2. **[2pt]** *Write the gradient of this objective with respect to θ (one line). Note that we do not take gradients through the parameters of the target network, $Q_{\theta'}$.*

3. **[2pt]** *In the original DQN paper, visual observations from Atari are preprocessed by stacking the last $n = 4$ observations to produce the input. Why would we do this? What are the tradeoffs of having a larger or smaller n? (we won't be doing this for this assignment)*

4. **[2pt]** *Explain why DQN can be seen as a combination of function approximation, bootstrapping, and off-policy learning.*

5. ***[6pt] Replay Buffer***

(a) **[2pt]** *What are some benefits of using a replay buffer?*

(b) **[2pt]** *Why is it important to randomly sample transitions from the replay buffer rather than using consecutive samples?*

(c) **[2pt]** *What could go wrong if the buffer is too small or too large?*

6. ***[4pt] Target Network***

(a) **[2pt]** *How does using a target network help stabilize training compared to directly using Q_θ for both prediction and target?*

(b) **[2pt]** *Recall that we update the target network every k number of steps and then copy the weights of Q_θ, $\theta' \leftarrow \theta$. What are the tradeoffs of updating too quickly or too slowly?*

Exercise 4. [30pt] Into the DQN-Verse

We will now implement a simple version of DQN and test it on some easy environments. The experiments in this assignment should take around 2 minutes each.

1. **[20pt]** *Implement the following functions in the provided code:*

   - *(functions to implement in* `learn.py`*)* `compute_DQN_loss`, `update_target`
   - *(functions to implement in* `model.py`*)* `QModel`*. This sets the architecture of Q_θ. Choose a simple architecture that allows you to get good results without taking too much time. The environments we use is solvable with simple architectures.*
   - *(functions to implement in* `schedule.py`*)* `LinearSchedule.update` *and* `ExplorationSchedule.get_action`*. These functions set up our learning rate scheduler, as well as a scheduler for annealing the ϵ in ϵ-greedy action selection (which means that with probability ϵ, we choose a random action, and with probability $(1-\epsilon)$ we choose the greedy action according to Q_θ. We want to start with a large value for ϵ to encourage exploration (note that in the code, we explore for* `learning_start` *number of steps before starting updates on Q_θ). We then gradually reduce exploration as our policy improves to stabilize training.*

2. **[10pt]** *Report Training rewards, Evaluation rewards, max Q, and Loss from the logged plots in* `results/` *on* `Acrobot-v1` *and* `CartPole-v0` *(you can set the environment in* `main.py`*.) (8 plots) The hyperparameters given in* `config` *are not tuned to be optimal and you may experiment with these to get better performance.Test your code with multiple random seeds to ensure the performance is consistent (you can set the random seed using* `seed_all()` *in* `main.py`*).*

Exercise 5. [10pt] Q-Network, Q-Network on the wall, which action is the BEST of them all? (Spoiler alert: it doesn't know!)

Your Q-network trained with DQN is overestimating the value of some actions. Don't panic! We can fix this. First, let's identify the cause of the overestimation in simple tabular Q-learning, and then we'll explore one way to resolve it.

Consider an MDP with three states S, M, and E:

- *From S, only action a is available; it leads deterministically to M with reward 0.*
- *From M, actions a_1, a_2, a_3 are available; each leads to E with reward $r \sim \mathcal{N}(0, \sigma^2)$.*
- *E is terminal.*

Update the notebook `Q_Learning_bias` as follows:

- *In the first cell of notebook, write the update rule of Q-Learning at states S and M.*
- *In the second cell, write the update rule of SARSA at states S and M.*

Answer the following questions:

1. ***[2pt]** Draw the scheme of the MDP implemented in the notebook provided.*
2. ***[2pt]** Before running the code, what do you expect to see as the Q values by following the policy implemented in the notebook?*
3. ***[2pt]** Report the outputs of Q-Learning and SARSA cells?*
4. ***[2pt]** What is the effect of decreasing/increasing number of runs and σ in the reward of actions at state M?*
5. ***[2pt]** Are the results the same as what you expected? If not, what could be the reason behind that?*

   (Hint: https://math.stackexchange.com/questions/89030/expectation-of-the-maximum-of-gaussian-random-variables)

Exercise 6. [20pt] Double DQN (DDQN)

Double DQN (DDQN) [Van Hasselt et al., 2016] is a variation of DQN to deal with the maximization bias issue of DQN. DDQN is the adaptation of Double Q-Learning to DQN. Instead of learning a separate Q network, we simply take advantage of the fact that we have two networks already: Q_θ and the target $Q_{\theta'}$. Our targets become:

$$Y = R + \gamma Q_{\theta'}\left(X', \underset{a' \in \mathcal{A}}{argmax}\, Q_\theta(X', a')\right)$$

1. **[10pt]** *Implement DDQN by filling in the function `compute_DoubleDQN_loss` in `learn.py`. Note that you can test this function by setting `double = True` in `main.py`.*

2. **[10pt]** *Produce and report the learning curves (Training rewards, Eval rewards, max Q, Loss) on* `Acrobot-v1` *and* `CartPole-v0` *for DDQN (8 plots). Do you notice a difference between DQN and DDQN? (It's fine if you don't!)*

Exercise 7. [5pt Bonus] Be the Next RLstar: Propose a New Way to Resolve Maximization Bias

Try to come up with an original solution to mitigate maximization bias. Describe the reasoning and thought process that led you to your idea. Some directions to think about:

- *How to decrease the bias of expectation over max of random variables?*
- *Other ways of getting more estimates instead of using the target network.*

Advice: *Please avoid asking LLMs for the answer. Sure, doing so might give you an* ***immediate reward*** *of 5 points, but in the long run, it won't help you become the next Rockstar of DRL. Remember the key lesson of this course: chasing immediate rewards is rarely the best strategy.*

References

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015. 4

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016. 6