# CS 446 / ECE 449 — Homework 6

*your NetID here*

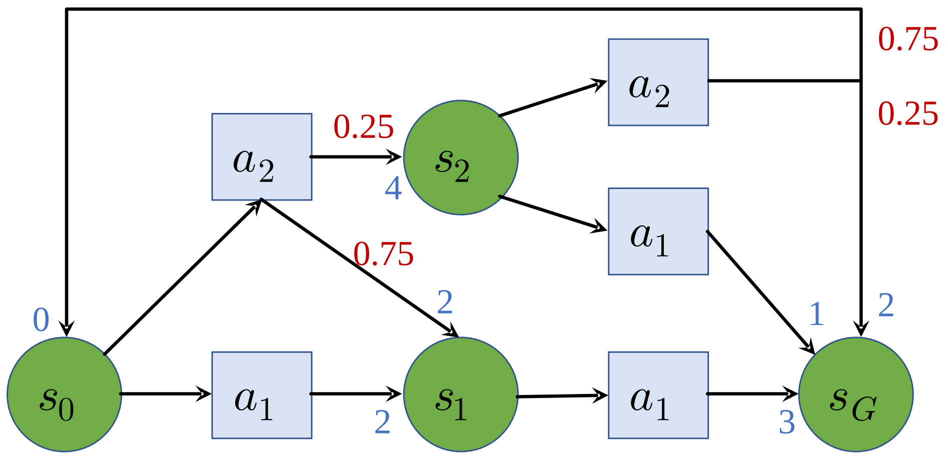
Version 1.3

**Instructions.**

- Homework is due **Wednesday, December 3, at noon CST**; you have **3** late days in total for **all Homeworks**.
- The template for Q3 and Q4 are available at this link.
- Everyone must submit individually at gradescope under `Homework 6` and `Homework 6 Code`.
- The "written" submission at `Homework 6` **must be typed**, and submitted in any format gradescope accepts (to be safe, submit a PDF). You may use LaTeX, markdown, google docs, MS word, whatever you like; but it must be typed!
- When submitting at `Homework 6`, gradescope will ask you to **mark out boxes around each of your answers**; please do this precisely!
- Please make sure your NetID is clear and large on the first page of the homework.
- Your solution **must** be written in your own words. Please see the course webpage for full **academic integrity** information. You should cite any external reference you use.
- We reserve the right to reduce the auto-graded score for `Homework 6 Code` if we detect funny business (e.g., your solution lacks any algorithm and hard-codes answers you obtained from someone else, or simply via trial-and-error with the autograder).
- When submitting to `Homework 6 Code`, only upload `hw6_q_learning.py` and `hw6_reinforce.py`. Additional files will be ignored.

## 1. Markov Decision Process

Consider the MDP presented in the figure below. States $\{s_1, s_2, s_G\}$ are represented by green circles. Actions $\{a_1, a_2\}$ are presented by blue squares. Transition probabilities are marked in red and rewards in blue.



(a) (1 point) Explain the difference between a deterministic and a stochastic Markov Decision Process. Is the given MDP stochastic or deterministic? Explain your answer.

(b) (1.5 points) What are three mechanisms to find the optimal policy $\pi^*$ for a given MDP?

(c) (3 points) For the policy $\pi(s_0) = a_1$, $\pi(s_1) = a_1$, what is the policy graph and the resulting value functions $V^\pi(s_1)$ and $V^\pi(s_0)$?

(d) (3 points) For the policy $\pi(s_0) = a_2, \pi(s_2) = a_1$, what is the policy graph and the resulting value functions $V^\pi(s_2)$, $V^\pi(s_1)$ and $V^\pi(s_0)$?

(e) (3 points) For the policy $\pi(s_0) = a_2$, $\pi(s_2) = a_2$, what is the policy graph and the resulting value functions $V^\pi(s_2)$, $V^\pi(s_1)$ and $V^\pi(s_0)$?

(f) (2 points) What is the optimal policy for the MDP given in Fig. 1? Briefly explain your answer.

**Solution.**

## 2. Q-Learning [Written]

(a) State the Bellman optimality principle as a function of the optimal Q-function $Q^*(s,a)$, the expected reward function $R(s,a,s')$ and the transition probability $P(s'|s,a)$, where $s$ is the current state, $s'$ is the next state and $a$ is the action taken in state $s$

(b) In case the transition probability $P(s'|s,a)$ and the expected reward $R(s,a,s')$ are unknown, a stochastic approach is used to approximate the optimal Q-function. After observing a transition of the form $(s,a,r,s')$, write down the update of the Q-function at the observed state-action pair $(s,a)$ as a function of the learning rate $\alpha$, the discount factor $\gamma$, $Q(s,a)$ and $Q(s',a')$.

(c) What is the advantage of an epsilon-greedy strategy?

(d) What is the advantage of using a replay-memory?

(e) Consider a system with two states $S_1$ and $S_2$ and two actions $a_1$ and $a_2$. You perform actions and observe the rewards and transitions listed below. Each step lists the current state, reward, action and resulting transition as: $S_i; R = r; a_k : S_i \rightarrow S_j$. Perform Q-learning using a learning rate of $\alpha = 0.5$ and a discount factor of $\gamma = 0.5$ for each step by applying the formula from part (b). The Q-table entries are initialized to zero. Fill in the tables below corresponding to the following four transitions. What is the optimal policy after having observed the four transitions?

i. $S_1$; $R = -10$; $a_1 : S_1 \rightarrow S_1$

ii. $S_1$; $R = -10$; $a_2 : S_1 \rightarrow S_2$

iii. $S_2$; $R = 18.5$; $a_1 : S_2 \rightarrow S_1$

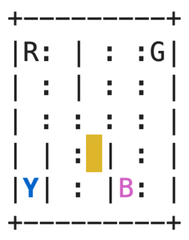
iv. $S_1$; $R = -10$; $a_2 : S_1 \rightarrow S_2$

**Solution.**

## 3. Q-Learning [Coding]

In this problem, you need to implement a *tabular* Q-learning algorithm to train an agent to play the game `Taxi-v3`.[1] The game is shown in the figure below. Your agent should be performing well after around 1000 episodes.

You can install gym via `pip install gym`.



```
+---------+
|R: | : :G|
| : | : : |
| : : : : |
| | :█| : |
|Y| : |B: |
+---------+
```

Please take a look at the game definition page to understand how to play the game.[2] Essentially, there is a single passenger needing a taxi delivery service. The agent controls the taxi and needs to pick up and drop off that passenger at the designated locations.

- There are six possible actions: `move south/north/east/west`, `pick up passenger`, and `drop off passenger`.
- There are four possible locations that the passenger may appear, namely `R`, `G`, `Y`, and `B`.
- There are four possible destinations, namely `R`, `G`, `Y`, and `B`. Destination and pick-up location will not be the same.
- Every episode, colored texts indicate the *actual* pick-up and drop-off locations. For example, in the snapshot, the agent (yellow rectangle) needs to pick up the passenger at the blue text (`Y`) and then drive to the pink text (`B`) for drop off. Note, colors are only used for visualization. When coding, you do not need to deal with colors since all states are encoded with numerics.
- Reward settings:
  - successfully deliver the passenger: +20;
  - incorrect pick-up or drop-off: -10;
  - to encourage fast task completion, there is a reward of -1 for each step.

(a) Implement action selection in evaluation mode. Corresponding function: `_act_eval` of class `Agent`.

(b) Implement $\epsilon$-greedy exploration strategy for action selection in training mode. Corresponding function: `_act_train` of class `Agent`.

**Hint:** you may find `env.action_space.sample()` useful.

(c) Implement Q-value update taught in lectures in function `update` of class `Agent`. You should update `q_table` of `Agent`.

(d) Plot the training curve. By default, the script will save a `train_moving_avg.png`. Attach it here.

(e) Attach the episode generated during evaluation to show the qualitative performance of your trained agent. You can just take a series of snapshots printed by the script.

**Solution.**

[1] `https://gym.openai.com/envs/Taxi-v3/`

[2] `https://github.com/openai/gym/blob/master/gym/envs/toy_text/taxi.py`

## 4. REINFORCE [Written + Coding]

In this problem we investigate the difference between maximum likelihood estimation (MLE) and reinforcement learning.

We are given a utility $U(\theta) = \mathbb{E}_{p_\theta}[R(y)] = \sum_{y\in\mathcal{Y}} p_\theta(y)R(y)$ which is the expected value of the (potentially) non-differentiable reward $R(y)$ defined over a discrete domain $y \in \mathcal{Y} = \{1, \ldots, |\mathcal{Y}|\}$. Our goal is to learn the parameters $\theta$ of a probability distribution $p_\theta(y)$ so as to obtain a high utility (high expected reward), *i.e.*, we want to find $\theta^* = \arg\max_\theta U(\theta)$. To this end we define the probability distribution to read

$$p_\theta(y) = \frac{\exp F_\theta(y)}{\sum_{\hat{y}\in\mathcal{Y}} \exp F_\theta(\hat{y})}, \tag{1}$$

where, $F_\theta(y) \in \mathbb{R}$ is a scoring function.

(a) If we are given a dataset $\mathcal{D}$ with i.i.d. samples, we can learn the parameters $\theta$ of a distribution via maximum likelihood, *i.e.*, by addressing

$$\max_\theta \sum_{y\in\mathcal{D}} \log p_\theta(y)$$

via gradient descent. State the cost function and its gradient when plugging the model specified in Eq. (1) into this program.

(b) If we aren't given a dataset but if we are instead given a reward function $R(y)$ we search for the parameters $\theta$ by maximizing the utility $U(\theta)$, *i.e.*, the expected reward. Explain how we can approximate the utility with $N$ samples from the probability distribution $p_\theta(y)$.

(c) Using general notation, what is the gradient of the utility $U(\theta)$ w.r.t. $\theta$, *i.e.*, what is $\nabla_\theta U(\theta)$. How can we approximate this value with $N$ samples $\{\tilde{y}_i\}_{i=1}^N$from $p_\theta(y)$? Make sure that you state the gradient in the form which ensures that computation via sampling from $p_\theta(y)$ is possible.

(d) Using the parametric probability distribution defined in Eq. (1), what is the approximated gradient of the utility (use $N$ samples from $p_\theta(y)$)? How is this gradient related to the result obtained in part (a)?

(e) In hw6_reinforce.py we compare the two forms of learning.

- Let the size of the domain $|\mathcal{Y}| = 6$, and let the groundtruth data distribution $p_{\text{GT}}(y) = 1/12$ for $y \in \{1, 6\}$, $p_{\text{GT}}(y) = 2/12$ for $y \in \{2, 5\}$, and $p_{\text{GT}}(y) = 3/12$ for $y = \{3, 4\}$.
- The dataset $\mathcal{D}$ in (a) contains $|\mathcal{D}| = 1000$ points sampled from this distribution.
- Further let $F_\theta(y) = [\theta]_y$, where $\theta \in \mathbb{R}^6$ and where $[a]_y$ returns the $y$-th entry of vector $a$. The reward function happens to equal the groundtruth distribution, *i.e.*, $R(y) = p_{\text{GT}}(y)$.

Complete hw6_reinforce.py:

i. In function `reinforce`, sample 1000 points from current step distribution $p_\theta$.
**Hint:** You may find `torch.multinomial` useful.

ii. Compute the gradient based on your answer in (d).

Answer the following questions:

i. What distribution $p_\theta$ is learned with the maximum likelihood approach? Paste the distribution here.
**Hint:** the script default prints `torch.nn.Softmax(dim=0)(theta_mle)`. You can directly paste the default printed information here.

ii. What distribution is learned with the REINFORCE approach? Paste the distribution here.
**Hint:** the script default prints `torch.nn.Softmax(dim=0)(theta_rl)`. You can directly paste the default printed information here.

iii. Explain why this is expected.

**Solution.**