0 Instructions (Total 40 pts)

Homework is due Tuesday, April 30, 2024 at 23:59pm Central Time. Please refer to `https://courses.grainger.illinois.edu/cs446/sp2024/homework/hw/index.html` for course policy on homeworks and submission instructions.

1 Bellman Equation (7 pts)

1. (1 pts) What is the definition of the Q value function, $Q^\pi(s,a)$, for an infinite horizon with discount γ, reward function $R(s,a)$, stochastic transition dynamics $p(s' \mid s,a)$, with policy $\pi(a \mid s)$?
2. (2 pts) Derive the Bellman equation from this definition.
3. (1 pts) Given Q and transition (s,a,r,s'), what is the Q-learning update step (for discount γ and learning rate α)?
4. (1 pts) If the maximum reward is $R_{max} = \max_{s,a} R(s,a)$, with discount factor γ, what is the maximum value of the optimal Q, $\max_{s,a} Q^*(s,a)$?
5. (2 pts) 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: $S_i; R = r; a_k : S_i \to S_j$. 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 four tables below corresponding to the following four transitions. What is the final policy after having observed the four transitions?
   (a) S_1; $R = -10$; $a_1 : S_1 \to S_1$
   (b) S_1; $R = -10$; $a_2 : S_1 \to S_2$
   (c) S_2; $R = 18.5$; $a_1 : S_2 \to S_1$
   (d) S_1; $R = -10$; $a_2 : S_1 \to S_2$

Solution:

1. $Q^\pi(s,a) = \sum_{i=0}^{\infty} \gamma^i R(s_i,a_i) \mid s_0 = s, a_0 = a, a_i \sim \pi(s_i), s_{i+1} \sim p(s_i,a_i)$
2. $Q^\pi(s,a) = \sum_{i=0}^{\infty} \gamma^i R(s_i,a_i) = R(s,a) + \sum_{i=1}^{\infty} \gamma^i R(s_i,a_i)$
   $= R(s,a) + \gamma \sum_{i=0}^{\infty} \gamma^i R(s_{i+1},a_{i+1})$
   $= R(s,a) + \gamma Q(s_1,a_1) = R(s,a) + \gamma \mathbb{E}_{s' \sim p(s,a), a' \sim \pi(s')} Q(s',a')$
3. $Q(s,a) = \alpha Q(s,a) + (1-\alpha)(r + \gamma \max_{a'} Q(s',a'))$
4. Optimal satisfies the Bellman equation so $Q_{max} \le R_{max} + \gamma Q_{max} \implies Q_{max} \le \frac{R_{max}}{1-\gamma}$
5. i.

Q	S_1	S_2
a_1	-5	.
a_2	.	.

   ii.

Q	S_1	S_2
a_1	-5	.
a_2	-5	.

   iii.

Q	S_1	S_2
a_1	-5	8
a_2	-5	.

iv.

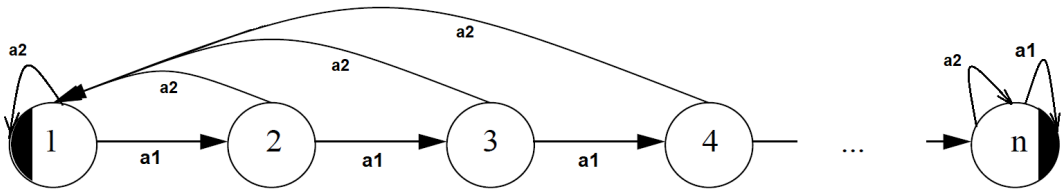
Q	S_1	S_2
a_1	-5	8
a_2	-5.5	.

Final policy: $\pi(a_1|S_1) = 1.0, \pi(a_1|S_2) = 1.0$

2 Combination Lock (7 pts)

Consider an MDP with n states $s_1, s_2, \ldots, s_n$, where there are 2 actions a_1, a_2 at each state. At each state s_i, $i < n$ action a_1 takes the agent to the s_{i+1} and for state s_n, action a_1 is a self-loop which keeps the agent in s_n. At any state, action a_2 takes the agent back to s_1, except for the last state, which again has a self-loop. See the figure for an overall picture of the setting. $R(s_i, a_j)$ (reward of action a_j at state s_i) is 0, except for (s_n, a_1), which has a value of 1. The agent takes one action at each step.

With *uniform random policy*, the agent at each state chooses one of the available actions uniformly at random. Now considering this *combination lock* MDP, answer the questions below. You need to show your work to receive full credit for each of the sections.



1. (2 pts) Compute the expected number of steps for the uniform random policy to go from state s_1 to state s_n.
2. (3 pts) Compute the formula for $Q(s_i, a_j)$, $\forall i, j$ for the uniform random policy considering a discounted reward setting with a discount factor of γ.
3. (2 pts) Now consider a new *greedy policy* using the $Q(s, a)$ function you computed in the previous part. Specifically, $\pi(s_i) = argmax_a Q(s_i, a)$ (i.e., the agent, at each state, chooses the action with the highest value of the Q function computed in part (b)). Compute the new expected number of steps to get from state s_1 to s_k. Hint: how does $Q(s_i, a_1)$ compare to $Q(s_i, a_2)$?

Solution:

1. Let l_i represent the expected time to reach step i. Then we have:

$$l_k = \frac{1}{2}(l_{k-1} + 1) + \frac{1}{2}(l_{k-1} + 1 + l_k) \implies l_k = 2l_{k-1} + 2 \implies l_n = 2^n - 2.$$

For the base case, we compute l_2:

$$l_2 = \frac{1}{2} + 2(\frac{1}{2})^2 + \dots$$
$$\implies \frac{1}{2}l_2 = (\frac{1}{2})^2 + 2(\frac{1}{2})^3 + \dots$$
$$\implies \frac{1}{2}l_2 = \frac{1}{2} + (\frac{1}{2})^2 + \dots = 1 \implies l_2 = 2$$

2. **NOTE: this solution from last year seems to assume you get reward for both a_1 and a_2 at s_n**
For the last state:

$$Q(s_n, a_2) = Q(s_n, a_1) = \sum_{k=0}^{\infty} \gamma^k = \frac{1}{1-\gamma}.$$

And for the one to last state and action a_1:

$$Q(s_{n-1}, a_1) = \frac{\gamma}{2}\left[Q(s_n, a_1) + Q(s_n, a_2)\right] = \frac{\gamma}{1-\gamma}.$$

For all the states except for the last one we have:

$$\forall i < n : Q(s_i, a_2) = \frac{\gamma}{2}\left[Q(s_1, a_2) + Q(s_1, a_1)\right] \implies Q(s_1, a_2)$$
$$= \frac{\gamma/2}{1-\frac{\gamma}{2}} Q(s_1, a_1) = \frac{\gamma}{2-\gamma} Q(s_1, a_1).$$

Therefore for state s_{n-2}, we have:

$$Q(s_{n-2}, a_1) = \frac{\gamma}{2}\left[Q(s_{n-1}, a_1) + Q(s_{n-1}, a_2)\right] = \frac{\gamma}{2}\left[\frac{\gamma}{1-\gamma} + \frac{\gamma}{2-\gamma} Q(s_1, a_1)\right]$$
$$= \frac{1}{2}\frac{\gamma^2}{1-\gamma} + \frac{1}{2}\frac{\gamma^2}{2-\gamma} Q(s_1, a_1)$$

And the general pattern for action a_1 appears to be:

$$Q(s_{n-i}, a_1) = (\frac{1}{2})^{i-1}\frac{\gamma^i}{1-\gamma} + \left(\sum_{j=1}^{i-1}(\frac{1}{2})^j \frac{\gamma^{j+1}}{2-\gamma}\right) Q(s_1, a_1)$$

Writing this for s_1 ($i = n-1$), we get:

$$\implies Q(s_1, a_1) = (\frac{1}{2})^{n-2}\frac{\gamma^{n-1}}{1-\gamma} + \left(\sum_{j=1}^{n-2}(\frac{1}{2})^j \frac{\gamma^{j+1}}{2-\gamma}\right) Q(s_1, a_1)$$
$$\implies Q(s_1, a_1) = \frac{(\frac{1}{2})^{n-2}\frac{\gamma^{n-1}}{1-\gamma}}{1 - \frac{\gamma}{2-\gamma}\frac{1-(\gamma/2)^{n-1}}{1-\gamma/2}}$$

3. Because in each state s_i, $Q(s_i, a_1) > Q(s_i, a_2)$, the greedy policy always chooses action a_1 and goes to the next state. So it takes $n-1$ steps for the agent to go from s_1 to s_n.

3 Q-value initialization (7 pts)

In this question we will explore the effect of initialization on Q-learning. Let $Q_1(s,a) = 0, Q_2(s,a) = R_{max}$, where $R_{max} = \max_{s,a} R(s,a)$. Consider the greedy policy $\pi_i(s) = \text{argmax}_a\, Q_i(s,a)$, where ties are broken at random (by selecting a random action among equal values).

Consider the MDP in the previous problem:

1. (2 pts) What is the expected number of steps for each policy to reach s_n starting from s_1?
2. (1 pts) Now assume that each time we explore a state transition (s, a, s', r) we update Q_i with a Bellman update, i.e.,

$$Q_i(s,a) = (1-\alpha)Q_i(s,a) + \alpha(r + \gamma \max_{a'}(Q_i(s',a')))$$

   for $0 < \gamma < 1, 0 < \alpha < 1$.

   Now what is the expected number of steps for π_1 to reach s_n starting from s_1?
3. (1 pts) Assuming we reset the agent (π_1) to s_1 once it takes action a_1 at s_n (and gets reward of 1), what is the expected number of steps before it reaches s_n again?
4. (1 pts) How can using a replay buffer help reduce the number if episodes before π_1 is optimal?
5. (2 pts) What is the minimum number of steps for π_2 to reach s_n starting from s_1 (again assuming we update Q_2 each step)?
6. (1 pt extra credit) Show that the maximum number of steps that π_2 takes to reach s_n is upper bound by your answer to 3.2 (the expected number of steps for π_1 to reach s_n). Assuming we update the Q values each step.

Solution:

1. Since all values are the same both policies are uniform random, so the answer is same as 2(a).
2. Again, the same, because every update has no effect until we reach the goal.
3. So far we have only updated the value of a single entry in the table, $Q(s_n, a_1)$, and so the answer is the same, i.e., $l_n = 2^n - 2$
4. Using a replay buffer means we store (for example) transitions like $(s_{n-2}, a_1, r = 0, s_{n-1})$ and could sample this transition in order to update values of other states based on our newly computed value for (s_{n-1}, a_1).
5. Min is $n-1$, can just get lucky.
6. Max: Each time we explore an edge the value of that transition will go down since the reward is 0 and the initial value is larger. In the worst case every edge (s_i, a_1) will be taken the same number of times as (s_i, a_2) for all i. The length of the longest path in which this happens has (s_{n-1}, a_1) and (s_{n-1}, a_2) once, $(s_{n-2}, a_{1,2})$ twice, etc. We can write a recurrence as before $l_k = 2l_{k-1} + 2$, the same value as in part 2.

4 Policy Gradient (8 pts)

In policy gradient methods our goal is to evaluate the policy π directly rather than the value function. Let $J(\pi)$ be the expected value of the policy for some initial state distribution d_0. In practice, π is a parameterized function, say π_θ.

1. (3 pts) For trajectory $\tau = s_0, a_0, r_0, ..., s_T, a_T, r_T$ the discounted sum of rewards is $R(\tau) = \sum_{i=0}^{T} \gamma^i r_i$ and $P^\pi(\tau)$ is the probability of τ under the policy, i.e., $P^\pi(\tau) = d_0(s_0) \prod_{i=0}^{T-1} \pi(a_i \mid s_i) P(s_{i+1} \mid s_i, a_i) \pi(a_T \mid s_T)$. Show that

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[R(\tau) \sum_{i=0}^{T} \nabla_\theta \log \pi_\theta(a_i \mid s_i) \right]$$

2. (3 pts) An algorithm using the above gradient requires rolling out trajectories for each update. Let $d_i^\pi(s)$ be the probability the policy visits state s after i steps from the starting state distribution, and let $d^\pi(s)$ be the (discounted) probability that the policy visits s on some (infinite) trajectory, i.e., $d^\pi(s) = \sum_{i=0}^{\infty} \gamma^i d_i^\pi(s)$. Note this is called the stationary distribution, or discounted state occupancy. Show that

$$\nabla_\theta J(\pi_\theta) = \mathop{\mathbb{E}}_{s \sim d^\pi, a \sim \pi_\theta(s)} [Q^\pi(s, a) \nabla_\theta \log \pi_\theta(a \mid s)]$$

   Hint: Notice that $J(\pi) = \mathbb{E}_{s \sim d_0} [V^\pi(s)]$, where V is the value function, and so $\nabla_\theta J(\pi_\theta) = \nabla_\theta [\mathbb{E}_{s \sim d_0} [V^{\pi_\theta}(s)]]$. Now how do you write $V^{\pi_\theta}(s)$ in terms of Q^{π_θ}?

3. (2 pts) Finally, show that

$$\nabla_\theta J(\pi_\theta) = \mathop{\mathbb{E}}_{s \sim d^\pi, a \sim \pi_\theta(s)} [(Q^\pi(s, a) - f(s)) \nabla_\theta \log \pi_\theta(a \mid s)]$$

   for any function f. Many types of policy gradient depend on appropriate choice of f to reduce the variance of the estimated gradient.

Solution:

1.

$$\nabla_\theta J(\pi_\theta) = \nabla_\theta \mathbb{E}_{\tau \sim \pi_\theta} [R(\tau)] = \sum_\tau [R(\tau) \nabla_\theta P^\pi(\tau)] = \sum_\tau [R(\tau) P^\pi(\tau) \nabla_\theta \log P^\pi(\tau)]$$

$$= \sum_\tau \left[R(\tau) P^\pi(\tau) \nabla_\theta \left(\sum_{i=0}^{T} \log \pi(a_i \mid s_i) + \log d_0(s_0) + \sum_{i=0}^{T-1} \log P(s_{i+1} \mid s_i, a_i) \right) \right]$$

$$\sum_\tau \left[R(\tau) P^\pi(\tau) \nabla_\theta \left(\sum_{i=0}^{T} \log \pi(a_i \mid s_i) \right) \right] = \mathbb{E}_{\tau \sim \pi_\theta} \left[R(\tau) \sum_{i=0}^{T} \nabla_\theta \log \pi_\theta(a_i \mid s_i) \right]$$

2.

$$\nabla_\theta V^{\pi_\theta}(s) = \nabla \sum_a \pi(a \mid s) Q^\pi(s, a) = \sum_a [Q^\pi(s, a) \nabla \pi(a \mid s) + \pi(a \mid s) \nabla Q^\pi(s, a)]$$

$$= \sum_a \left[Q^\pi(s,a)\pi(a \mid s)\nabla \log \pi(a \mid s) + \pi(a \mid s)\nabla \left(R(s,a) + \gamma \mathbb{E}_{s' \sim P(s,a)} V^\pi(s') \right) \right]$$

$$= \mathbb{E}_{a \sim \pi} \left[Q^\pi(s,a)\nabla \log \pi(a \mid s) + \gamma \mathbb{E}_{s' \sim P(s,a)} \nabla V^\pi(s') \right]$$

Therefore,

$$\nabla_\theta J(\pi_\theta) = \nabla_\theta \left[\mathbb{E}_{s \sim d_0} \left[V^{\pi_\theta}(s) \right] \right]$$

$$= \mathbb{E}_{s \sim d_0, a \sim \pi(s)} \left[Q^\pi(s,a)\nabla \log \pi(a \mid s) + \gamma \mathbb{E}_{s' \sim P(s,a)} \nabla V^\pi(s') \right]$$

$$= \mathbb{E}_{s \sim d_0, a \sim \pi(s)} \left[Q^\pi(s,a)\nabla \log \pi(a \mid s) \right] + \gamma \mathbb{E}_{s \sim d_1^\pi} \nabla V^\pi(s')$$

$$= \sum_{i=0}^{\infty} \gamma^i \mathbb{E}_{s \sim d_i^\pi, a \sim \pi(s)} \left[Q^\pi(s,a)\nabla \log \pi(a \mid s) \right] = \mathop{\mathbb{E}}_{s \sim d^\pi, a \sim \pi(s)} \left[Q^\pi(s,a)\nabla \log \pi(a \mid s) \right]$$

3.

$$\mathop{\mathbb{E}}_{s \sim d^\pi, a \sim \pi_\theta(s)} \left[f(s)\nabla_\theta \log \pi_\theta(a \mid s) \right] = \mathop{\mathbb{E}}_{s \sim d^\pi} \left[f(s)\nabla_\theta \sum_a \pi(a \mid s) \log \pi_\theta(a \mid s) \right]$$

$$= \mathop{\mathbb{E}}_{s \sim d^\pi} \left[f(s) \sum_a \pi(a \mid s)\nabla_\theta \log \pi_\theta(a \mid s) \right] = \mathop{\mathbb{E}}_{s \sim d^\pi} \left[f(s) \sum_a \nabla_\theta \pi_\theta(a \mid s) \right]$$

$$= \mathop{\mathbb{E}}_{s \sim d^\pi} \left[f(s)\nabla_\theta \sum_a \pi_\theta(a \mid s) \right] = \mathop{\mathbb{E}}_{s \sim d^\pi} \left[f(s)\nabla_\theta(1) \right] = 0$$

5 Coding: Tabular Q-Learning (11 pts)

In this problem you will be implementing tabular Q learning for the Taxi-v3 environment.

1. (2 pts) Your first task is to investigate the OpenAI gym API and the Taxi-v3 environment. You can create the environment with

```
env = gym.make("Taxi-v3", render_mode="ansi")
```

   (a) What do states correspond to? How many possible states are there? What is the action space?

   (b) What do env.step() and env.reset() return? How are states represented?

   (c) What are the valid render_modes in gym.make? For Taxi-v3 what does each render mode do/return?

   (d) Go digging in the open source documentation - how does render() convert the state representation returned by step and reset (hint: self.s) to the human readable features you described in part (a). Your answer should be a function call, "self.function_name(self.s)".

2. (2 pts) Implement the QAgent class, more instructions are in the code. Please use np.argmax to get the best action (break ties in value by taking the first action). Report the train_rewards.png plot created by q_learning() for an experiment with $\alpha = 0.1, \gamma = 0.9, \epsilon = 0.1$ and 10,000 epochs.

3. (2 pts) What happens if you initialize the Q table values to -10 (the minimum reward in the environment)? Why does it perform worse? Report the train_rewards.png plot (with -10 initial Q values) created by q_learning() for an experiment with $\alpha = 0.1, \gamma = 0.9, \epsilon = 0.1$ and 10,000 epochs.

4. (1 pts) What percentage of the time does a uniform random policy ($\epsilon = 1.0$) get positive reward (success) in the environment across 10,000 epochs (printed for you as "Episode success rate")? Report the train_rewards.png plot for this uniform random policy.

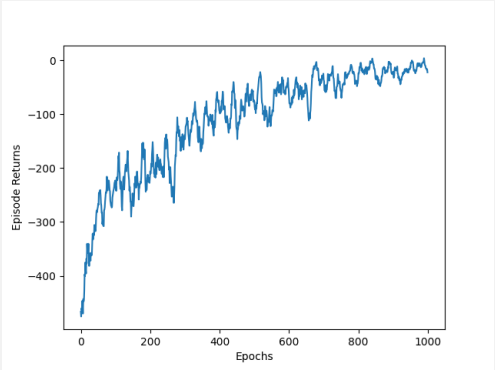
5. (1 pts) With initialization of -10, report the train_rewards.png plot created by q_learning() for an experiment with $\alpha = 0.1, \gamma = 0.9$, 10000 epochs and ϵ of your choice which yields a model that successfully passes the evaluation (meaning the taxi delivers the person to the goal in the call to "eval_agent()").

6. (2 pts) Some actions lead to "no-op", for example moving right when there is a wall to the right. In the documentation for the Taxi-v3 environment they describe a method to only sample actions which lead to a change in state. Implement this; now, what percentage of the time does a uniform random policy ($\epsilon = 1.0$) get positive reward (success) in the environment across 10,000 epochs? Report the train_rewards.png plot.

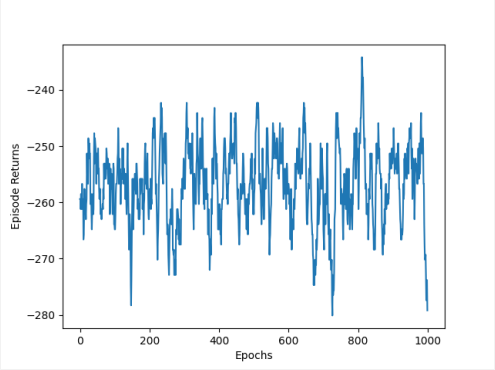
7. (1 pts) Again, with initialization of -10 AND never taking an action that leads to "no-op", report the train_rewards.png plot created by q_learning() for an experiment with $\alpha = 0.1, \gamma =$ 0.9, 10000 epochs and ϵ *the same as in part 5* which yields a model that successfully passes the evaluation.

Solution:

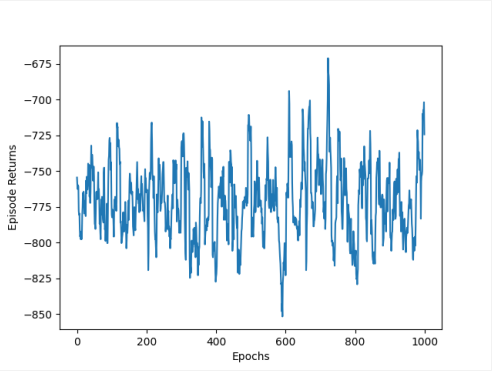
1. (a) Taxi pos, agent pos (or in taxi), agent goal. 500 states, only 404 are reachable. 6 actions - 4 directions and pickup/dropoff.
   (b) step returns (next_state, reward, terminated, truncated, info). reset returns (state, info). States are represented as integers.
   (c) Render modes are human (which makes a colorful rendering of the environment and displays it each *step()*), rgb_array (render() returns the image as an array), and ansi (render() returns a string representation, letters are goal locations which are colored according to person start/goal, taxi is green square).
   (d) self.decode(self.s), which does some mod and division on the integer state.
2. Normal:



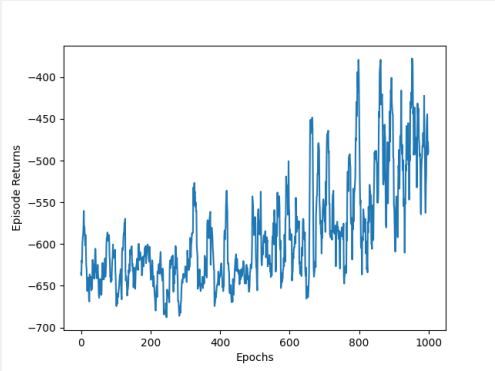
3. Negative 10 initialization:
   Initializing to zero is optimistic init, which as we saw in problem 3 leads to better exploration properties (trying more transitions).



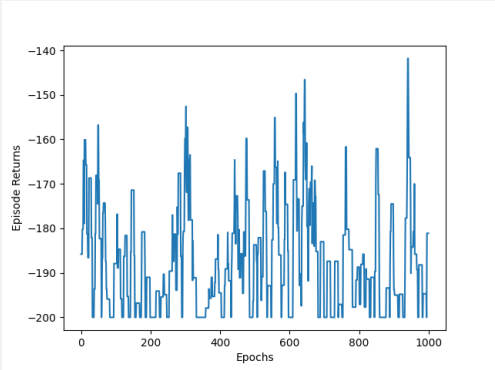
4. Uniform random. Success rate: 0.044



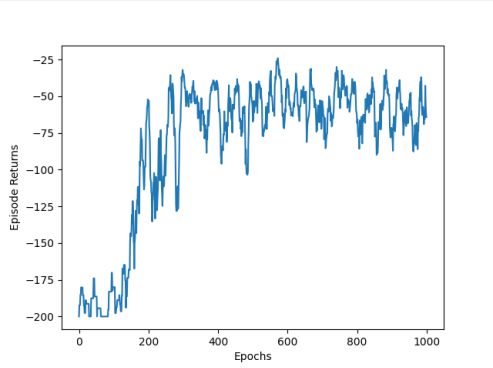
5. Negative 10, $\epsilon = 0.75$. Success rate: 0.82. Note they don't need to report success rate here.



6. Uniform random, action masked. Success: 0.133



7. Init -10, $\epsilon = 0.75$, action masked. Success: 0.96. (Note they don't need to report success rate here.)


Episode Returns
−25
−50
−75
−100
−125
−150
−175
−200
0
200
400
600
800
1000
Epochs