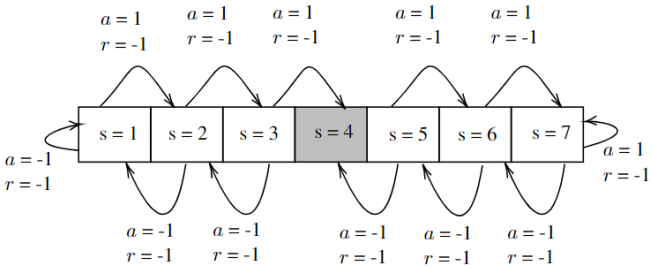

i. (5 points) Let us perform tabular Q-learning on this chain MDP. Suppose we observe the following ordered 4-step trajectory; each sample in the trajectory is of the form (*state*, *action*, *reward*):

$$(3, -1, -1), (2, 1, -1), (3, 1, -1), (4, 1, 0)$$

Initializing all Q-values to 0, give updated values for $Q(3, -1)$, $Q(2, 1)$, and $Q(3, 1)$. Use learning rate $\alpha = 0.5$.

Solution:

Assuming discount rate γ = 1 $\quad Q(s,a) := Q(s,a) + \alpha\,[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$

Randomly sampling trajectory, assuming we get (4,1,0) and then update:

Q(4, 1) = 0

Randomly sampling trajectory, assuming we get (3,1,0) and then update:

Q(3, 1) = Q(3, 1) +0.5[-1 + 1 * 0 – Q(3, 1)]

= 0 + 0.5 * [-1 + 1 * 0 – 0]

= - 0.5

Randomly sampling trajectory, assuming we get (2,1,0) and then update:

Q(2, 1) = Q(2, 1) + 0.5[-1 + 1 * Q(3, -1) – Q(3, 1)]

= 0 + 0.5 * [-1 + 1 * 0 – 0]

= - 0.5

Randomly sampling trajectory, assuming we get (3,-1,0) and then update:

Q(3, -1) = Q(3, -1) + 0.5[-1 + 1 * Q(2, -1) – Q(3, 1)]

= 0 + 0.5 * [-1 + 1 * 0 – 0]

= - 0.5

Continuing sampling and updating …

ii. (10 points) Let us approximate the Q-function as:

$$\hat{q}(s, a; \boldsymbol{w}) = \boldsymbol{w}^T \cdot \begin{bmatrix} s \\ a \\ 1 \end{bmatrix} = w_0 \cdot s + w_1 \cdot a + w_2$$

Given the parameters $\boldsymbol{w}$ and a single sample (s, a, r', s'), the loss function we will minimize is

$$J(\boldsymbol{w}) = (r' + \gamma \max_{a'} \hat{q}(s', a'; \boldsymbol{w}^-) - \hat{q}(s, a; \boldsymbol{w}))^2$$

where $\hat{q}(s', a'; \boldsymbol{w}^-)$ is a target network parameterized by fixed weights $\boldsymbol{w}^-$.
Now, suppose we currently have weight vectors:

$\boldsymbol{w} = \begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix}$ and $\boldsymbol{w}^- = \begin{bmatrix} 1 \\ -1 \\ -2 \end{bmatrix}$

and we observe a sample $(s, a, r', s') = (2, -1, -1, 1)$.
Perform a *single* gradient update of the parameters $\boldsymbol{w}$ given this sample. Use learning rate $\alpha = 0.25$. Write out the gradient $\nabla_{\boldsymbol{w}} J(\boldsymbol{w})$ and new parameters $\boldsymbol{w}'$.

Assuming discount rate $\gamma = 1$

$\nabla_w J(w) = 2\hat{q}(s, a; w) * [s\ a\ 1]^T - 2(r' + \gamma \max_{a'} \hat{q}(s' : a' : w^-)) * [s\ a\ 1]^T$

$= 2 * \hat{q}(2, -1; w) * [2\ -1\ 1]^T - 2(-1 + 1 * \hat{q}(1, -1; w^-)) * [2\ -1\ 1]^T$

$= 2 * (-2) * [2\ -1\ 1]^T - 2(-1 + 0) * [2\ -1\ 1]^T$

$= [-4\ \ 2\ \ -2]^T$

$w' = \mathrm{w} + \alpha * \nabla_w J(w) = [-1\ \ 1\ \ 1]^T + 0.25 * [-4\ \ 2\ \ -2]^T = [-2\ \ \ 1.5\ \ \ 0.5]^T$

(b) (5 points) Is the REINFORCE algorithm guaranteed to converge to the optimal policy if the policy class can represent the optimal policy? Explain briefly.

As a stochastic gradient method, REINFORCE algorithm has good theoretical convergence properties. By construction, the expected update over an episode is in the same direction as the performance gradient. This assures an improvement in expected performance for sufficiently small a, and convergence to a local optimum under standard stochastic approximation conditions for decreasing a.

(c) (20 points) Consider the 4×4 Grid world shown below that is littered with obstacles (shaded cells). It has a special goal state ($s = 10$) such that entering the goal state gives a reward $r = 1$ and taking any action from the goal state ends the episode with a reward of $r = 0$. The world also has a special punishment state ($s = 3$) such that entering this undesired state gives a reward $r = -1$ and taking any action from the state ends the episode with a reward of $r = 0$. All actions are deterministic.

s=11	s=12		s=13
s=7	s=8	s=9	s=10 +1
s=4		s=5	s=6
s=1	s=2	s=3 -1	

Define the function $h(s, a; \boldsymbol{\theta})$, which gives a parameterized quantitative preference for each state-action pair, as:

$$h(s, a; \boldsymbol{\theta}) = \boldsymbol{\theta}^T \cdot \boldsymbol{x}(s, a) = \theta_1 \cdot x_1(s, a) + \theta_2 \cdot x_2(s, a) + \theta_3 \cdot x_3(s, a).$$

i. (5 points) Formulate the policy in the form of a soft max in action preferences. Derive its gradient w.r.t. $\boldsymbol{\theta}$, i.e., give $\nabla_{\boldsymbol{\theta}}\, \pi(a|s; \boldsymbol{\theta})$.

$$\pi(a|s; \theta) = \frac{e^{h(s,a;\theta)}}{\sum_{a' \in A} e^{h(s,a';\theta)}}$$

$$\nabla_\theta \pi(a|s; \theta) = \frac{e^{h(s,a;\theta)} * \boldsymbol{x}(s, a) * (\sum_{a' \in A} e^{h(s,a';\theta)} - e^{h(s,a;\theta)})}{(\sum_{a' \in A} e^{h(s,a';\theta)})^2}$$

ii. (15 points) Let the feature function

- $x_1(s, a)$ give the length in Manhattan distance of the shortest path from s' to the *punishment state* where s' is the state that results on performing action a from state s.
- $x_2(s, a)$ gives 1 divided by the length in Manhattan distance of the shortest path from s' to the *goal state* where s' is the state that results on performing action a from state s.
- $x_3(s, a)$ is 0.33 if performing action a from state s leads to an obstacle otherwise it is 1.

Consider an episode (12,→,-0.04), (12,↓,-0.04), (8,→,-0.04), (9,→,1), (10,→,0) generated by an initial policy $\pi(a|s;\boldsymbol{\theta})$.

Show the updates of $\boldsymbol{\theta}_t$ as Monte Carlo REINFORCE goes through each step of the episode. You may initialize $\boldsymbol{\theta}^T = [1\ \ 1\ \ 1]$. Show your work.

REINFORCE: Monte-Carlo Policy-Gradient Control (episodic) for π_*

Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta})$
Algorithm parameter: step size $\alpha > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ (e.g., to $\mathbf{0}$)

Loop forever (for each episode):
  Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot, \boldsymbol{\theta})$
  Loop for each step of the episode $t = 0, 1, \ldots, T-1$:
    $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$ (G_t)
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\gamma^t G \nabla \ln \pi(A_t|S_t, \boldsymbol{\theta})$

$$\nabla_\theta \ln \pi(a|s;\theta) = \ln\left(\frac{e^{h(s,a;\theta)} * \boldsymbol{x}(s,a) * \left(\sum_{a' \in A} e^{h(s,a';\theta)} - e^{h(s,a;\theta)}\right)}{\left(\sum_{a' \in A} e^{h(s,a';\theta)}\right)^2}\right)$$

$$= \frac{h(s,a;\theta) + \boldsymbol{x}(s,a) + ln\left(\sum_{a' \in A} e^{h(s,a';\theta)} - e^{h(s,a;\theta)}\right)}{2 \ln \sum_{a' \in A} e^{h(s,a';\theta)}}$$

Assuming learning rate α = 0.25 and discount rate γ = 1

First Loop (12,→,-0.04):

G = -0.04

$$\boldsymbol{x}(s,a) = \boldsymbol{x}(12, \rightarrow) = [4\ \ 0.33\ \ 0.33]^\top$$

$$h(12,\rightarrow;\theta) = \boldsymbol{\theta} * \boldsymbol{x}(12,\rightarrow) = [4\ \ 0.33\ \ 0.33]^{\top}$$

$$e^{h(12,\rightarrow;\theta)} = [e^{4}\ \ e^{0.33}\ \ e^{0.33}]^{\top}$$

$$\sum_{a' \in A} e^{h(s,a';\theta)} = [e^{5} + 2e^{4} + e^{3} \qquad 2e^{0.33} + e^{0.5} + e^{0.25} \qquad 2e^{0.33} + 2e\ ]^{\top}$$

$$\theta = [111]^{\top} + 0.25 * (-0.04 * \frac{h(s,a;\theta) + \boldsymbol{x}(s,a) + ln\left(\sum_{a' \in A} e^{h(s,a';\theta)} - e^{h(12,\rightarrow;\theta)}\right)}{2 \ln \sum_{a' \in A} e^{h(s,a';\theta)}})$$

$$= [111]^{\top} + 0.25 * (-0.04 * \frac{[4\ \ 0.33\ \ 0.33]^{\top} + [4\ \ 0.33\ \ 0.33]^{\top} + ln\left([e^{5}+e^{4}+e^{3} \qquad e^{0.33}+e^{0.5}+e^{0.25} \qquad e^{0.33}+2e\ ]^{\top}\right)}{2 \ln([e^{5}+2e^{4}+e^{3} \qquad 2e^{0.33}+e^{0.5}+e^{0.25} \qquad 2e^{0.33}+2e\ ]^{\top})})$$

$$= [0.988 \qquad 0.994 \qquad 0.994]^{\top}$$

Second LOOP (12,→,-0.04), (12,↓,-0.04):

G = - 0.04 – 1 * 0.04 = - 0.08

$$\boldsymbol{x}(s,a) = \boldsymbol{x}(12,\downarrow) = [3\ \ 0.5\ \ 1]^{\top}$$

$$\boldsymbol{x}(12,\rightarrow) = [4\ \ 0.33\ \ 0.33]^{\top}$$

$$\boldsymbol{x}(12,\uparrow) = [4\ \ 0.33\ \ 0.33]^{\top}$$

$$\boldsymbol{x}(12,\leftarrow) = [5\ \ 0.25\ \ 1]^{\top}$$

$$h(12,\downarrow;\theta) = \boldsymbol{\theta} * \boldsymbol{x}(12,\downarrow) = [0.988 \quad 0.994 \quad 0.994]^{\top} * [3\ \ 0.5\ \ 1]^{\top} = [2.994 \quad 0.497 \quad 0.994]^{\top}$$

$$e^{h(12,\downarrow;\theta)} = [e^{2.994}\ \ e^{0.497}\ \ e^{0.994}]^{\top}$$

$$e^{h(12,\rightarrow;\theta)} = [\ ]^{\top}$$

$$e^{h(12,\uparrow;\theta)} = [\ ]^{\top}$$

$$e^{h(12,\leftarrow;\theta)} = [\ ]^{\top}$$

$$\sum_{a' \in A} e^{h(s,a';\theta)} = [\ ]^{\top}$$

$$\theta = [111]^{\top} + 0.25 * (-0.04 * \frac{h(s,a;\theta) + \boldsymbol{x}(s,a) + ln\left(\sum_{a' \in A} e^{h(s,a';\theta)} - e^{h(12,\downarrow;\theta)}\right)}{2 \ln \sum_{a' \in A} e^{h(s,a';\theta)}})$$

$$= [111]^{\top} + 0.25 * (-0.04 * \frac{[2.994 \quad 0.497 \quad 0.994]^{\top} + [3\ \ 0.5\ \ 1]^{\top} + ln\left([\ ]^{\top} - [e^{2.994} \quad e^{0.497} \quad e^{0.994}]^{\top}\right)}{2 \ln([\ ]^{\top})})$$

$$= [\ ]^{\top}$$

I already do all the details for one loop, then what we need do is to follow the procedures. Therefore, remaining computation would become worthless and I just skip that.

And here is a **bug** in the designing of the feature function. Specially, if we focus on x2(s, a) (gives **1 divided by the length** in Manhattan distance of the shortest path from s′ to the goal state). For trajectories (9, →, 1) and (10, →, 0), the value of x2(9, →) and x2(10, →) will become infinite or undefined due to **1 divide by 0**. As a result, the second part of θ parameter will become infinite number.

(d) (5 points) Briefly discuss one advantage that maximum entropy methods have over the maximum margin family of methods for inverse RL. Also give one clear limitation of optimizing the maximum entropy over trajectories. Be as specific as possible.

Advantage: By framing problems of imitation learning as solutions to Markov Decision Problems, maximum entropy methods reduce learning to the problem of recovering a utility function that makes the behavior induced by a near-optimal policy closely mimic demonstrated behavior.

Limitation: By using a small feature space, the maximum entropy methods can not incorporate contextual factors or other special features (e.g., time of day, weather) into our feature space (like inducing region-based or even specific road-based features that can explain, e.g., the avoidance of a particular trajectory).