ECE59500RL Homework 5

Robert (Cars) Chandler — chandl71@purdue.edu

Problem 1

1.1

The objective function for ERM with square loss is:

$$\min_{f \in \mathcal{F}} \sum_{i=1}^{N} \|\hat{y} - y\|^2$$

And in this case, $\hat{y} = Q_\theta$ which comes from a class of functions parameterized by the four θ values:

$$\min_{f \in \mathcal{F}} \sum_{i=1}^{N} \|\left(\theta_1 s_i^2 + \theta_2 a_i^2 + \theta_3 s_i a_i + \theta_4\right) - y\|^2$$

So we want to find the $Q \in \mathcal{Q}$ that yields the least squared error, which is equivalent to finding $\theta \in \mathbb{R}^4$, the parameterization of $Q \in \mathcal{Q}$ yielding the least squared error:

$$\hat{Q}^{\pi_t} = \arg\min_{\theta \in \mathbb{R}^4} \sum_{i=1}^{N} \|\left(\theta_1 s_i^2 + \theta_2 a_i^2 + \theta_3 s_i a_i + \theta_4\right) - y\|^2$$

We are given three iterations of data ($N = 3$) which we can substitute into this form to give the final result of the objective function:

$$
\begin{aligned}
&\arg\min_{\theta\in\mathbb{R}^4} \sum_{i=1}^{N} \|\Big(\theta_1 s_i^2 + \theta_2 a_i^2 + \theta_3 s_i a_i + \theta_4\Big) - y\|^2 \\
&\arg\min_{\theta\in\mathbb{R}^4} \Big[\Big(\theta_1 \cdot 1^2 + \theta_2 \cdot 0^2 + \theta_3 \cdot 1 \cdot 0 + \theta_4 - 1\Big)^2 + \\
&\quad \Big(\theta_1 \cdot (-2)^2 + \theta_2 \cdot 1^2 + \theta_3 \cdot -2 \cdot 1 + \theta_4 - 0\Big)^2 + \\
&\quad \Big(\theta_1 \cdot 1^2 + \theta_2 \cdot (-1)^2 + \theta_3 \cdot 1 \cdot -1 + \theta_4 - 3\Big)^2\Big] \\
&\arg\min_{\theta\in\mathbb{R}^4} \Big[(\theta_1 + \theta_4 - 1)^2 + (4\theta_1 + \theta_2 - 2\theta_3 + \theta_4)^2 + (\theta_1 + \theta_2 - \theta_3 + \theta_4 - 3)^2\Big]
\end{aligned}
$$

1.2

$$\pi_{t+1}(a|s) = (1-\alpha)\pi_t(a|s) + \alpha\bar{\pi}(a|s)$$

We know that π_0 is a uniform distribution across the action space:

$$\pi_0(a|s) = \left[\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right], \quad \forall s \in \mathcal{S}$$

Next, we need to calculate $\bar{\pi}$ according to:

$$\bar{\pi}(s) = \arg\max_{a\in\mathcal{A}} \hat{Q}^{\pi_t}(s,a), \forall s \in \mathcal{S}$$

We can iterate over combinations of the relevant subset of the state-action space to determine which action maximizes the quantity above and then calculate the updated policy according to the definition above:

```python
import itertools

import numpy as np
import xarray as xr
from IPython.display import Markdown

state_space = np.arange(-3, 4)
action_space = np.arange(-1, 2)


def qhat0(s, a):
    return 2 * s**2 + a**2 - s * a + 0.5
```

```python
# Initialize qhat0
q_eval = xr.DataArray(
    data=np.zeros([len(state_space), len(action_space)]),
    coords={"s": state_space, "a": action_space},
)

# Evaluate qhat0 at all (s, a)
for s, a in itertools.product(state_space, action_space):
    q_eval.loc[dict(s=s, a=a)] = qhat0(s, a)

# Get the optimal actions for qhat0
i_optimal_action = q_eval.argmax(dim="a").to_numpy()  # type: ignore

optimal_actions = action_space[i_optimal_action]

# Uniform distribution
p0 = xr.full_like(q_eval, 1 / 3)

pbar = xr.zeros_like(q_eval)

# We calculated optimal actions, so we "pick" the action accordingly (assign it
# probability 1)
for s, a in zip(state_space, optimal_actions):
    pbar.loc[dict(s=s, a=a)] = 1

alpha = 0.25

# Convex combination of policies
p1 = (1 - alpha) * p0 + alpha * pbar

# Get the distributions for s = 1, 2
p1_s1 = Markdown(str(p1.sel(s=1).data.tolist()))
p1_s2 = Markdown(str(p1.sel(s=2).data.tolist()))
```

So the probability distribution of action values (in the same order they appear in the action space definition) are:

$$\pi_1(a|1) = [0.5, 0.25, 0.25]$$
$$\pi_1(a|2) = [0.5, 0.25, 0.25]$$

Problem 2

2.1

For $s \in \mathcal{S} \in \tau$:

$$a^* = \arg\max_{a \in \mathcal{A}} \hat{Q}(s, a)$$

$$\begin{aligned} a^*(s = b) &= x \\ a^*(s = c) &= x \\ a^*(s = d) &= x \\ a^*(s = e) &= y \end{aligned}$$

For $a \in \mathcal{A}$:

$$\pi(a|s) = 1 - \varepsilon + \frac{\varepsilon}{|\mathcal{A}|} \text{ if } a = a^* \text{ else } \frac{\varepsilon}{|\mathcal{A}|}$$

$$\begin{aligned} \pi(x|b) &= 1 - 0.1 + \frac{0.1}{2} = 0.95 & \pi(y|b) &= \frac{0.1}{2} = 0.05 \\ \pi(x|c) &= 1 - 0.1 + \frac{0.1}{2} = 0.95 & \pi(y|c) &= \frac{0.1}{2} = 0.05 \\ \pi(x|d) &= 1 - 0.1 + \frac{0.1}{2} = 0.95 & \pi(y|d) &= \frac{0.1}{2} = 0.05 \\ \pi(x|e) &= \frac{0.1}{2} = 0.05 & \pi(y|e) &= 1 - 0.1 + \frac{0.1}{2} = 0.95 \end{aligned}$$

2.2

The dataset of three points is formed by three (x_i, y_i) pairs where $x_i = (s_i, a_i)$ and

$$y_i = r_i + \gamma \max_{a' \in \mathcal{A}} \hat{Q}(s_{i+1}, a')$$

The x_i are trivial to form, and we can calculate the y_i here:

$$\begin{aligned}
y_t &= r_t + 0.9 \max_{a' \in \mathcal{A}} \hat{Q}(s_{t+1}, a') \\
&= 1 + 0.9 \cdot 0.75 \\
&= 1.675
\end{aligned}$$

$$\begin{aligned}
y_{t+1} &= r_{t+1} + 0.9 \max_{a' \in \mathcal{A}} \hat{Q}(s_{t+2}, a') \\
&= -2 + 0.9 \cdot -1.5 \\
&= -3.35
\end{aligned}$$

$$\begin{aligned}
y_{t+2} &= r_{t+2} + 0.9 \max_{a' \in \mathcal{A}} \hat{Q}(s_{t+3}, a') \\
&= -0.5 + 0.9 \cdot -1.5 \\
&= -1.85
\end{aligned}$$

So the dataset is:

$$\{(x_t = (c, x), y_t = 1.675), (x_{t+1} = (e, x), y_{t+1} = -3.35), (x_{t+2} = (b, y), y_{t+2} = -1.85)\}$$

2.3

The objective function for ERM with square loss is:

$$\min_{f \in \mathcal{F}} \sum_{i=1}^{N} \|\hat{\mathbf{y}} - \mathbf{y}\|^2$$

where $\hat{\mathbf{y}} = \boldsymbol{\theta}^\top \boldsymbol{\phi}(s, a)$. We can calculate $\hat{y}$ for each data point:

For $x_t = (c, x)$:

$$\hat{y}_t = \begin{bmatrix} \theta_1 & \theta_2 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \end{bmatrix} = \theta_2 - \theta_1$$

For $x_{t+1} = (e, x)$:

$$\hat{y}_t = \begin{bmatrix} \theta_1 & \theta_2 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \theta_2$$

For $x_{t+2} = (b, y)$:

$$\hat{y}_t = \begin{bmatrix}\theta_1 & \theta_2\end{bmatrix}\begin{bmatrix}-2\\-1\end{bmatrix} = -2\theta_1 - \theta_2$$

So using these with our values for y from the dataset, the objective function becomes:

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^2} \sum_{i=1}^{N} \|\hat{\mathbf{y}}_i - \mathbf{y}_i\|^2$$
$$= \min_{\boldsymbol{\theta}\in\mathbb{R}^2} \left[(\theta_2 - \theta_1 - 1.675)^2 + (\theta_2 + 3.35)^2 + (-2\theta_1 - \theta_2 + 1.85)^2\right]$$

Note: if we want to find the actual θ parameters, we would just replace the min above with an arg min.

2.4

We can solve this by forming a simple design matrix from the feature vectors we have and then finding the least squares solution using that with the output vector we have from the y_i values:

```python
import numpy as np
from IPython.display import Markdown

a = np.array([[-1, 1], [0, 1], [-2, -1]])
b = np.array([1.675, -3.35, -1.85])
thetas, sum_sq_err, *_ = np.linalg.lstsq(a, b)

t1, t2 = [Markdown(f"{theta:.5f}") for theta in thetas]
sse = Markdown(f"{sum_sq_err.item():.5f}")
```

$$\theta_1 = 0.42143$$
$$\theta_2 = -0.08214$$

and the result of the objective function (the minimal sum of square errors) is 16.61161.

Problem 3

3.1

To compute $\nabla_\theta \log \pi_\theta(a_t|s_t)$ for the softmax policy:

$$\pi_\theta(a|s) = \frac{\exp(\theta_{s,a})}{\sum_{a' \in \mathcal{A}} \exp(\theta_{s,a'})}$$

We can split up the numerator and denominator using logarithm properties and solve the gradient of the resulting difference:

$$\begin{aligned}\nabla_\theta \log \pi_\theta(a|s) &= \nabla_\theta \left(\theta_{s,a} - \log \sum_{a' \in \mathcal{A}} \exp(\theta_{s,a'}) \right) \\ &= \nabla_\theta \theta_{s,a} - \frac{\nabla_\theta \sum_{a' \in \mathcal{A}} \exp(\theta_{s,a'})}{\sum_{a' \in \mathcal{A}} \exp(\theta_{s,a'})}\end{aligned}$$

This becomes a difference of vectors where each element in the vector is equal to the result above where the gradient operator becomes a partial derivative operator with respect to each of the possible $\theta_{s,a}$ values. For example, the entries look like:

$$\frac{\partial}{\partial \theta_{s_i,a_j}} \theta_{s,a} - \frac{\frac{\partial}{\partial \theta_{s_i,a_j}} \sum_{a' \in \mathcal{A}} \exp(\theta_{s,a'})}{\sum_{a' \in \mathcal{A}} \exp(\theta_{s,a'})}, \quad \forall s_i \in \mathcal{S},\ \forall a_j \in \mathcal{A}$$

so the first term evaluates to:

$$\frac{\partial}{\partial \theta_{s_i,a_j}} \theta_{s,a} = \begin{cases} 1, & (s,a) = (s_i, a_j) \\ 0, & \text{otherwise} \end{cases}$$

The second term becomes

$$\frac{\frac{\partial}{\partial \theta_{s_i,a_j}} \sum_{a' \in \mathcal{A}} \exp(\theta_{s,a'})}{\sum_{a' \in \mathcal{A}} \exp(\theta_{s,a'})} = \begin{cases} \frac{\exp(\theta_{s_i,a_j})}{\sum_{a' \in \mathcal{A}} \exp(\theta_{s,a'})}, & s = s_i \\ 0, \quad \text{otherwise} \end{cases}$$

This is because the numerator expands out to a summation of $\exp(\theta_{s,a_k})$ terms, the partial derivative of which will evaluate to 0 if $s \neq s_i$ or $a_k \neq a_i$. The s_i corresponds to any of the indices of the vector where $s = s_i$, but the a_k corresponds to indices of the summation. So, for each entry in the vector, if the s matches the s which corresponds to the $\theta_{s,a}$ term with respect to which we are differentiating in that index, then the result will be nonzero, otherwise it will be zero. Once inside a nonzero index, then we are looking at terms of the summation inside that nonzero result, and only the term corresponding to the action with which respect to which we are differentiating will remain.

So, the first term will be nonzero only for a single index where we differentiate with respect to the correct θ. The second term will be present for all indices where we differentiate with respect to θ values with matching s values.

We can write these in terms of an indicator function using the notation $\mathbf{1}_{(s_i,a_j)}(s,a)$ where this is defined to be zero in all indices except for the index where $(s,a) = (s_i,a_j)$ corresponding to the parameter θ_{s_i,a_j}. The gradient vector is then comprised of the following expression for each state-action pair:

$$\nabla_\theta \log \pi_\theta(a|s) = \mathbf{1}_{(s_i,a_j)}(s,a) - \mathbf{1}_{s_i}(s)\frac{\exp(\theta_{s_i,a_j})}{\sum_{a'\in\mathcal{A}}\exp(\theta_{s_i,a'})}, \quad \forall s_i \in \mathcal{S},\ \forall a_j \in \mathcal{A}$$

So, if we need to evaluate this at a_t, s_t like the problem states, it becomes:

$$\nabla_\theta \log \pi_\theta(a_t|s_t) = \mathbf{1}_{(s_i,a_j)}(s_t,a_t) - \mathbf{1}_{s_i}(s_t)\frac{\exp(\theta_{s_i,a_j})}{\sum_{a'\in\mathcal{A}}\exp(\theta_{s_i,a'})}, \quad \forall s_i \in \mathcal{S},\ \forall a_j \in \mathcal{A}$$

At this point, we note that the final fraction in this expression is equivalent to the definition of the original softmax policy, so we can substitute this in, and our final answer is:

$$\nabla_\theta \log \pi_\theta(a_t|s_t) = \mathbf{1}_{(s_i,a_j)}(s_t,a_t) - \mathbf{1}_{s_i}(s_t)\pi_\theta(a_i|s_j), \quad \forall s_i \in \mathcal{S},\ \forall a_j \in \mathcal{A}$$

3.2

For softmax linear policies, we follow a similar line of reasoning to 3.1:

$$\begin{aligned}
\nabla_\theta \log \pi_\theta(a|s) &= \nabla_\theta \left(\boldsymbol{\theta}^\top \boldsymbol{\phi}(s,a) - \log \sum_{a'\in\mathcal{A}} \exp\left(\boldsymbol{\theta}^\top \boldsymbol{\phi}(s,a')\right)\right) \\
\nabla_\theta \log \pi_\theta(a|s) &= \nabla_\theta \left(\boldsymbol{\theta}^\top \boldsymbol{\phi}(s,a)\right) - \frac{\nabla_\theta \sum_{a'\in\mathcal{A}} \exp\left(\boldsymbol{\theta}^\top \boldsymbol{\phi}(s,a')\right)}{\sum_{a'\in\mathcal{A}} \exp\left(\boldsymbol{\theta}^\top \boldsymbol{\phi}(s,a')\right)} \\
\nabla_\theta \log \pi_\theta(a|s) &= \boldsymbol{\phi}(s,a) - \frac{\sum_{a'\in\mathcal{A}} \boldsymbol{\phi}(s,a') \exp\left(\boldsymbol{\theta}^\top \boldsymbol{\phi}(s,a')\right)}{\sum_{a'\in\mathcal{A}} \exp\left(\boldsymbol{\theta}^\top \boldsymbol{\phi}(s,a')\right)} \\
\nabla_\theta \log \pi_\theta(a|s) &= \boldsymbol{\phi}(s,a) - \sum_{a'\in\mathcal{A}} \pi_\theta(a'|s)\boldsymbol{\phi}(s,a')
\end{aligned}$$

And at (s_t, a_t), this becomes:

$$\nabla_\theta \log \pi_\theta(a_t|s_t) = \boldsymbol{\phi}(s_t, a_t) - \sum_{a' \in \mathcal{A}} \pi_\theta(a'|s_t)\boldsymbol{\phi}(s_t, a')$$

3.3

For softmax neural policies:

$$\begin{aligned}
\nabla_\theta \log \pi_\theta(a|s) &= \nabla_\theta \left(f_\theta(s,a) - \log \sum_{a' \in \mathcal{A}} \exp\left(f_\theta(s,a)\right) \right) \\
\nabla_\theta \log \pi_\theta(a\frac{\partial}{\partial \theta_{s_i,a_j}}\left(f_\theta(s,a')\right)|s) &= \nabla_\theta \left(f_\theta(s,a)\right) - \frac{\nabla_\theta \sum_{a' \in \mathcal{A}} \exp\left(f_\theta(s,a')\right)}{\sum_{a' \in \mathcal{A}} \exp\left(f_\theta(s,a')\right)} \\
\nabla_\theta \log \pi_\theta(a|s) &= \frac{\partial}{\partial \theta_{s_i,a_j}} f_\theta(s,a) - \frac{\sum_{a' \in \mathcal{A}} \frac{\partial}{\partial \theta_{s_i,a_j}}\left(f_\theta(s,a')\right)\exp\left(f_\theta(s,a')\right)}{\sum_{a' \in \mathcal{A}} \exp\left(f_\theta(s,a')\right)}, \quad \forall s_i \in \mathcal{S},\ \forall a_j \in \mathcal{A} \\
\nabla_\theta \log \pi_\theta(a|s) &= \frac{\partial}{\partial \theta_{s_i,a_j}} f_\theta(s,a) - \sum_{a' \in \mathcal{A}} \frac{\partial}{\partial \theta_{s_i,a_j}}\left(f_\theta(s,a')\right)\pi_\theta(a'|s), \quad \forall s_i \in \mathcal{S},\ \forall a_j \in \mathcal{A}
\end{aligned}$$

And at (s_t, a_t), this becomes:

$$\nabla_\theta \log \pi_\theta(a_t|s_t) = \frac{\partial}{\partial \theta_{s_i,a_j}} f_\theta(s_t,a_t) - \sum_{a' \in \mathcal{A}} \frac{\partial}{\partial \theta_{s_i,a_j}}\left(f_\theta(s_t,a')\right)\pi_\theta(a'|s_t), \quad \forall s_i \in \mathcal{S},\ \forall a_j \in \mathcal{A}$$

3.4

The A2C algorithm estimates the gradient by:

$$g_t = \frac{1}{1-\gamma} \nabla_\theta \log \pi_{\theta_t}(a_t|s_t) A^{\pi_{\theta_t}}(s_t, a_t)$$

and then updates θ accordingly:

$$\theta_{t+1} = \theta_t + \eta g_t$$

We already calculated the gradient term in 3.1 as:

$$\nabla_\theta \log \pi_\theta(a_t|s_t) = \mathbf{1}_{(s_i,a_j)}(s_t,a_t) - \mathbf{1}_{s_i}(s_t)\pi_\theta(a_i|s_j), \quad \forall s_i \in \mathcal{S},\ \forall a_j \in \mathcal{A}$$

3.4.a, 3.4.b

We are told that $(s_t, a_t) = (3, b)$, so the ∇_θ term will be zero for the indices corresponding to $\theta_{1,d}, \theta_{2,b}$ since neither the state nor the action match (s_t, a_t). As a result, the gradient estimate will be zero, and zero will be added in the update step, so **the parameters $\theta_{1,d}$ and $\theta_{2,b}$ will both *remain the same* after updating the parameters via stochastic gradient ascent.**

3.4.c

For $\theta_{3,b}$, both indicator functions are nonzero, so the element in its index evaluates to:

$$\frac{\partial}{\partial \theta_{3,b}} \log \pi_\theta(b|3) = 1 - \pi_\theta(b|3)$$

Therefore, the g_t term will be

$$\frac{1}{1-\gamma}(1 - \pi_\theta(b|3))A^{\pi_{\theta_t}}(s_t = 3, a_t = b)$$

The $\frac{1}{1-\gamma}$ term is a positive constant since we are in the discounted setting, $1-\pi_\theta(b|3)$ is positive since the softmax policy is comprised of an exponential divided by a sum of exponentials, yielding a positive quotient. Lastly, the $A^{\pi_{\theta_t}}(s_t, a_t)$ term is less than zero as given by the problem statement. So, g_t is the product of a two positive terms and a negative term, making it negative. So, a negative term (multiplied by some positive learning rate η) will be added to the current θ parameter, so **the parameter $\theta_{3,b}$ will *decrease* after the update step**.

3.4.d

For $\theta_{3,c}$, only the second indicator function is nonzero in the gradient since the state matches (s_t, a_t) but not the action, so the element in its index of the gradient vector evaluates to:

$$\frac{\partial}{\partial \theta_{3,c}} \log \pi_\theta(b|3) = -\pi_\theta(b|3)$$

Therefore, the g_t term will be

$$g_t = \frac{1}{1-\gamma} \cdot -\pi_\theta(a_t|s_t)A^{\pi_{\theta_t}}(s_t, a_t)$$

Here, we have a product of a positive γ term, a negative policy term, and a negative action term, yielding a positive gradient estimate, which will be added to the current θ parameter in the update step. Thus, **the parameter $\theta_{3,c}$ will *increase* after the update step**.

Problem 4

4.1

$$\mathcal{L}(\pi, \beta) = \mathbb{E}_{A \sim \pi(\cdot|s)} \left[Q^{\pi_t}(s, A) \right] + \lambda \mathcal{H}(\pi(\cdot|s)) - \beta \left(\sum_{a \in \mathcal{A}} \pi(a|s) - 1 \right)$$

We expand each term into its respective summation and differentiate:

$$\frac{\partial \mathcal{L}}{\partial \pi(a|s)} = \frac{\partial}{\partial \pi(a|s)} \left[\sum_{a' \in \mathcal{A}} \pi(a'|s) Q^{\pi_t}(s, a') - \lambda \sum_{a' \in \mathcal{A}} \pi(a'|s) \log \pi(a'|s) - \beta \left(\sum_{a' \in \mathcal{A}} \pi(a'|s) - 1 \right) \right]$$

The differentiation can be explained as follows: the first term is a summation of policy terms across each action, and the partial derivative will only evaluate to a nonzero value when $a' = a$, at which point it differentiates to one, so we end up just getting the constant Q from that summation. Similarly, the second summation only evaluates to a nonzero value when $a' = a$, but this time we use the product rule to evaluate:

$$\frac{\partial}{\partial \pi(a|s)} \pi(a|s) \log \pi(a|s) = \frac{\pi(a|s)}{\pi(a|s)} + \log \pi(a|s) \cdot 1 = 1 + \log \pi(a|s)$$

Lastly, the final summation evaluates to 1 when $a' = a$ and at this point, we just get the constant β as a result of our differentiation since the subtracted 1 is a constant eliminated by the differentiation.

$$\frac{\partial \mathcal{L}}{\partial \pi(a|s)} = \frac{\partial}{\partial \pi(a|s)} \left[\sum_{a' \in \mathcal{A}} \pi(a'|s) Q^{\pi_t}(s, a') - \lambda \sum_{a' \in \mathcal{A}} \pi(a'|s) \log \pi(a'|s) - \beta \left(\sum_{a' \in \mathcal{A}} \pi(a'|s) - 1 \right) \right]$$
$$\frac{\partial \mathcal{L}}{\partial \pi(a|s)} = Q^{\pi_t}(s, a) - \lambda \left(1 + \log \pi(a|s)\right) - \beta, \quad \forall a \in \mathcal{A}$$

4.2

The derivative with respect to β is much simpler as every term except the last summation cancels out:

$$\frac{\partial \mathcal{L}}{\partial \beta} = \frac{\partial}{\partial \beta}\left[\sum_{a'\in\mathcal{A}} \pi(a'|s) Q^{\pi_t}(s,a') + \lambda \sum_{a'\in\mathcal{A}} \pi(a'|s)\log\pi(a'|s) - \beta\left(\sum_{a'\in\mathcal{A}}\pi(a'|s) - 1\right)\right]$$
$$\frac{\partial \mathcal{L}}{\partial \beta} = 1 - \sum_{a'\in\mathcal{A}} \pi(a'|s)$$

4.3

Beginning with $\frac{\partial \mathcal{L}}{\partial \pi(a|s)} = 0$:

$$\begin{aligned} 0 &= Q^{\pi_t}(s,a) - \lambda\left(1 + \log\pi(a|s)\right) - \beta \\ \lambda\left(1+\log\pi(a|s)\right) &= Q^{\pi_t}(s,a) - \beta \\ \log\pi(a|s) &= \frac{Q^{\pi_t}(s,a)-\beta}{\lambda} - 1 \\ \pi(a|s) &= \exp\left(\frac{Q^{\pi_t}(s,a)-\beta}{\lambda} - 1\right) \end{aligned}$$

We now look at $\frac{\partial \mathcal{L}}{\partial \beta} = 0$:

$$\begin{aligned} 0 &= 1 - \sum_{a'\in\mathcal{A}}\pi(a'|s) \\ \sum_{a'\in\mathcal{A}}\pi(a'|s) &= 1 \end{aligned}$$

We can use this to continue solving for π where we previously left off:

$$\begin{aligned} \sum_{a'\in\mathcal{A}}\pi(a'|s) &= \sum_{a'\in\mathcal{A}}\exp\left(\frac{Q^{\pi_t}(s,a')-\beta}{\lambda} - 1\right) \\ 1 &= \sum_{a'\in\mathcal{A}}\exp\left(\frac{Q^{\pi_t}(s,a')-\beta}{\lambda}\right)e^{-1} \\ e &= \sum_{a'\in\mathcal{A}}\exp\left(\frac{Q^{\pi_t}(s,a')-\beta}{\lambda}\right) \\ e &= \sum_{a'\in\mathcal{A}}\exp\left(\frac{Q^{\pi_t}(s,a')}{\lambda}\right)e^{-\beta/\lambda} \\ \exp\left(\frac{\beta}{\lambda}+1\right) &= \sum_{a'\in\mathcal{A}}\exp\left(\frac{Q^{\pi_t}(s,a')}{\lambda}\right) \end{aligned}$$

Now we can substitute this back into our equation for $\pi(a|s)$ after splitting it into a product of exponentials:

$$\pi(a|s) = \exp\left(\frac{Q^{\pi_t}(s,a)}{\lambda}\right)\exp\left(-\frac{\beta}{\lambda}-1\right)$$
$$\pi(a|s) = \frac{\exp\left(\frac{Q^{\pi_t}(s,a)}{\lambda}\right)}{\sum_{a'\in\mathcal{A}}\exp\left(\frac{Q^{\pi_t}(s,a')}{\lambda}\right)}$$

This is the final solution for the stationary point of the Lagrangian. It takes the same form as the softmax policy function that we saw in Problem 3.

4.4

$$V^{\pi_{t+1}}(s) = \mathbb{E}_{a\sim\pi_{t+1}(\cdot|s)}\left[Q^{\pi_t}(s,a)\right] + \lambda\mathcal{H}(\pi_{t+1}(\cdot|s)), \quad \forall s\in\mathcal{S}$$

Expanding the entropy term:

$$\begin{aligned}
\lambda\mathcal{H}(\pi_{t+1}(\cdot|s)) &= -\lambda\sum_{a\in\mathcal{A}}\pi_{t+1}(a|s)\log\pi_{t+1}(a|s)\\
&= -\lambda\sum_{a\in\mathcal{A}}\pi_{t+1}(a|s)\log\frac{\exp\left(\frac{Q^{\pi_t}(s,a)}{\lambda}\right)}{\sum_{a'\in\mathcal{A}}\exp\left(\frac{Q^{\pi_t}(s,a')}{\lambda}\right)}\\
&= -\lambda\sum_{a\in\mathcal{A}}\pi_{t+1}(a|s)\left(\frac{Q^{\pi_t}(s,a)}{\lambda} - \log\sum_{a'\in\mathcal{A}}\exp\left(\frac{Q^{\pi_t}(s,a')}{\lambda}\right)\right)\\
&= -\lambda\left(\frac{\mathbb{E}_{a\sim\pi_{t+1}(\cdot|s)}\left[Q^{\pi_t}(s,a)\right]}{\lambda} - \log\sum_{a'\in\mathcal{A}}\exp\left(\frac{Q^{\pi_t}(s,a')}{\lambda}\right)\right)\\
&= -\mathbb{E}_{a\sim\pi_{t+1}(\cdot|s)}\left[Q^{\pi_t}(s,a)\right] + \lambda\log\sum_{a'\in\mathcal{A}}\exp\left(\frac{Q^{\pi_t}(s,a')}{\lambda}\right)
\end{aligned}$$

Substituting this back into the original equation:

$$\begin{aligned}
V^{\pi_{t+1}}(s) &= \mathbb{E}_{a\sim\pi_{t+1}(\cdot|s)}\left[Q^{\pi_t}(s,a)\right] + \lambda\mathcal{H}(\pi_{t+1}(\cdot|s)), \quad \forall s\in\mathcal{S}\\
&= \mathbb{E}_{a\sim\pi_{t+1}(\cdot|s)}\left[Q^{\pi_t}(s,a)\right] + -\mathbb{E}_{a\sim\pi_{t+1}(\cdot|s)}\left[Q^{\pi_t}(s,a)\right] + \lambda\log\sum_{a'\in\mathcal{A}}\exp\left(\frac{Q^{\pi_t}(s,a')}{\lambda}\right), \quad \forall s\in\mathcal{S}\\
&= \lambda\log\sum_{a\in\mathcal{A}}\exp\left(\frac{Q^{\pi_t}(s,a)}{\lambda}\right), \quad \forall s\in\mathcal{S}
\end{aligned}$$

Problem 5

5.1

We can represent the reward function and trajectories as data structures and compute the discounted return for each sub-trajectory in Python:

```python
import numpy as np
import xarray as xr
from IPython.display import Markdown

reward = xr.DataArray(
    data=np.array(
        [
            [-1, 1, 0, 0, 0],
            [-2, 0, 0, 3, -5],
            [-2, -2, 5, 0, 0],
            [-2, -2, 4, 0, 0],
        ]
    ),
    coords=dict(
        state=["empty", "monster", "food", "resource"],
        action=["stay", "explore", "collect", "evade", "befriend"],
    ),
)

trajectories = [
    [("empty", "stay"), ("monster", "evade"), ("resource", "collect")],
    [("empty", "explore"), ("food", "collect"), ("monster", "befriend")],
    [("empty", "explore"), ("empty", "explore"), ("resource", "collect")],
]

gamma = 0.95


returns = np.sum(
    np.array(
        [
            [
                gamma**t * reward.sel(state=s, action=a).item()
                for t, (s, a) in enumerate(traj)
            ]
```

```
                for traj in trajectories
            ]
        ),
        axis=1,
)

r1, r2, r3 = returns

return_latex = Markdown(
    "\n".join([rf"\tau_{i+1}&: {r:0.4f} \\" for i, r in enumerate(returns)])
)

pref_1_over_2 = np.exp(r1) / np.sum([np.exp(r1), np.exp(r2)])
pref_1_over_3 = np.exp(r1) / np.sum([np.exp(r1), np.exp(r3)])
```

Note: we also calculate the human preference probabilities at the end of this code, used in the next two parts.

The discounted returns for each sub-trajectory are as follows:

$$\begin{aligned}\tau_1 &: 5.4600 \\ \tau_2 &: 1.2375 \\ \tau_3 &: 5.5600\end{aligned}$$

5.2

The values are calculated at the end of the code block in 5.1 according to the following equation:

$$\mathbb{P}[\tau_1 > \tau_2] = \frac{\exp\left(G(\tau_1)\right)}{\exp\left(G(\tau_1)\right) + \exp\left(G(\tau_2)\right)}$$

where r_{τ_i} is the cumulative discounted return for τ_i as calculated in 5.1.

According to $\hat{R}$, the probability the human chooses τ_1 over τ_2 is:

$$\mathbb{P}[\tau_1 > \tau_2] = \frac{\exp\left(5.4600\right)}{\exp\left(5.4600\right) + \exp\left(1.2375\right)} = 0.98555$$

5.3

Using the same formula, the probability the human chooses τ_1 over τ_3 is:

$$\mathbb{P}[\tau_1 > \tau_3] = \frac{\exp(5.4600)}{\exp(5.4600) + \exp(5.5600)} = 0.47502$$

5.4

The MLE loss is calculated by summing log-probability terms for each point in the dataset.

We can use probability values calculated in the previous parts to fill in these variables. We calculate the loss in Python:

```python
import numpy as np
from p5_1 import pref_1_over_2, pref_1_over_3

pref_3_over_1 = 1 - pref_1_over_3

mle_loss = -(np.log(pref_1_over_2) + np.log(pref_3_over_1))
```

The MLE loss according to $\hat{R}$ is 0.6590.

5.5

If the human picks τ_2 over τ_1, then this is switching to a trajectory with a lower return than in 5.4. If it picks τ_3 over τ_1, this is the same choice made in 5.4, so this choice has no effect on the change in loss.

If switching from τ_2 to τ_1 means picking a trajectory with a lower return than in 5.4, then this means **the loss will increase**. This can be explained intuitively by the fact that choosing a worse return should increase the loss value. Mathematically, the probability of choosing the lower return will be smaller, which will cause the log-probability to decrease, causing the overall loss to increase since it negates this term.