PSET 3

Anusha Murali **CS182** **November 5, 2023**

Problem 1 Markov Decision Processes

(1) In the problem statement, the Bellman update operator B, which takes an estimate of the utility function $\hat{U}$ as input and returns a new utility estimate $B\hat{U} : S \to \mathbb{R}$, is defined for each $s \in S$ as

$$B\hat{U}(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s' \in S} P(s' \mid s, a)\hat{U}(s').$$

Let us consider any two utility estimates U', U'' of state s. We can then write:

$$\begin{aligned}
|BU'(s) - BU''(s)| &= \Bigg| R(s) + \gamma \max_{a \in A(s)} \sum_{s' \in S} P(s' \mid s, a)U'(s') \\
&\qquad - R(s) - \gamma \max_{a \in A(s)} \sum_{s' \in S} P(s' \mid s, a)U''(s') \Bigg| \\
&= \left| \gamma \max_{a \in A(s)} \sum_{s' \in S} P(s' \mid s, a)U'(s') - \gamma \max_{a \in A(s)} \sum_{s' \in S} P(s' \mid s, a)U''(s') \right| \\
&\leq \gamma \max_{a \in A(s)} \left| \sum_{s' \in S} P(s' \mid s, a)U'(s') - \sum_{s' \in S} P(s' \mid s, a)U''(s') \right|
\end{aligned}$$

The last step above uses the hint that $|\max_x f(x) - \max_x g(x)| \leq \max_x |f(x) - g(x)|$. Therefore, letting a^* as the most optimal action from state s, we can now write,

$$\begin{aligned}
|BU'(s) - BU''(s)| &\leq \gamma \max_{a \in A(s)} \left| \sum_{s' \in S} P(s' \mid s, a)U'(s') - \sum_{s' \in S} P(s' \mid s, a)U''(s') \right| \\
&= \gamma \left| \sum_{s' \in S} P(s' \mid s, a^*)U'(s') - \sum_{s' \in S} P(s' \mid s, a^*)U''(s') \right| \\
&= \gamma \left| \sum_{s' \in S} P(s' \mid s, a^*) \left(U'(s') - U''(s')\right) \right|
\end{aligned}$$

Therefore, since $U'(s) = \sum_{s' \in S} P(s' \mid s, a^*)U'(s')$ and $U''(s) = \sum_{s' \in S} P(s' \mid s, a^*)U''(s')$,

$$\begin{aligned}
\max_{s \in S} |BU'(s) - BU''(s)| &\leq \gamma \max_{s \in S} \left| \sum_{s' \in S} P(s' \mid s, a^*) \left(U'(s') - U''(s')\right) \right| \\
&\leq \gamma \max_{s \in S} |U'(s) - U''(s)| .
\end{aligned}$$

Hence, we find that, $\max_{s \in S} |BU'(s) - BU''(s)| \leq \gamma \max_{s \in S} |U'(s) - U''(s)|$. □

PSET 3

Anusha Murali **CS182** **November 5, 2023**

(2) We use a property called **contraction** to show that the value iteration converges. Contraction is a function of one argument that, when applied to two different inputs in turn, produces two output values that are "closer together," by at least some constant factor, than the original inputs. The following are the two important properties of contraction:

(a) A contraction has only one fixed point; if there were two fixed points they would not get closer together when the function was applied, so it would not be a contraction.

(b) When the function is applied to any argument, the value must get closer to the fixed point (because the fixed point does not move), so repeated application of a contraction always reaches the fixed point in the limit.

From the results from the previous part, we have,

$$\max_{s\in S} |BU'(s) - BU''(s)| \leq \gamma \max_{s\in S} |U'(s) - U''(s)|.$$

Therefore, the Bellman update is a contraction by a factor of γ on the space of utility vectors. Letting U' in the above inequality by the true utility U, so $BU = U$, we can write the results from Part 1 as,

$$\max_{s\in S} |U(s) - BU'(s)| \leq \gamma \max_{s\in S} |U(s) - U'(s)|.$$

If we view $\max_{s\in S} |U(s) - U'(s)|$ as the error in the estimate U', we see that the error is reduced by a factor of at least γ ($0 \leq \gamma < 1$) on each iteration, $t = 0, 1, 2, \ldots$. Let T be the number of iterations to reach an error of at most ϵ. Since the the utilities of all states are bounded by $\sum_{t=0}^{\infty} \gamma^t R_{\max} = |R_{\max}/(1-\gamma)|$, the maximum initial error is,

$$\max |U - U_0| \leq \frac{2R_{\max}}{(1-\gamma)}.$$

Since the error is reduced by a factor of γ on each iterations, we find that

$$\gamma^T \cdot \frac{2R_{\max}}{(1-\gamma)} \leq \epsilon.$$

Taking logs and finding the ceiling, we find the value of the number of iterations T, which is required to reach an error of at most ϵ as,

$$T = \left\lceil \frac{\log\left(\frac{2R_{\max}}{\epsilon(1-\gamma)}\right)}{\log(1/\gamma)} \right\rceil.$$

Due to the contraction property, for all $t \geq T$, the error will be $\leq \epsilon$. Therefore, if $\max_{s\in S} |U_{t+1} - U_t| < \epsilon(1-\gamma)/\gamma$, then there exists a $T \in \mathbb{N}$ such that $\max |U - U_{t+1}| < \epsilon$, where,

$$T = \left\lceil \frac{\log\left(\frac{2R_{\max}}{\epsilon(1-\gamma)}\right)}{\log(1/\gamma)} \right\rceil. \quad \square$$

PSET 3

Anusha Murali **CS182** **November 5, 2023**

Problem 2

(1) We formalize the problem into an MDP as follows:

- **States**: There are three states where Liz can be found, namely (1) Safely Employed (SE), (2) Performance Improvement Plan (PIP), and (3) Unemployed (UE). We use 0, 1, and 2 to denote SE, PIP and UE respectively in the Python code.
- **Actions**: There are two actions that Liz can take from the states SE and PIP, namely (1) Code and (2) Netflix. We use 0 and 1 to denote Code and Netflix actions respectively in the Python code.
- **Transition Functions**: Following are the transition functions for Liz taking the two different actions from the above states:

$$\begin{aligned} P(\text{SE}|\text{SE}, \text{Code}) &= 1 \\ P(\text{SE}|\text{SE}, \text{Netflix}) &= \frac{1}{4} \\ P(\text{PIP}|\text{SE}, \text{Netflix}) &= \frac{3}{4} \\ P(\text{SE}|\text{PIP}, \text{Code}) &= \frac{1}{4} \\ P(\text{PIP}|\text{PIP}, \text{Code}) &= \frac{3}{4} \\ P(\text{SE}|\text{PIP}, \text{Netflix}) &= \frac{7}{8} \\ P(\text{UE}|\text{PIP}, \text{Netflix}) &= \frac{1}{8} \end{aligned}$$

  All other transition probabilities are zero.
- **Rewards**: Each day when Liz decides to watch Netflix, she gets $R(s) = +10$ utility. If she decides to code, she gets $R(s) = +4$ utility.

(2) Let π_1 be policy that Liz always watches Netflix. Therefore we obtain the value function for policy π_1 as,

$$U(s) = \sum_{s'} P\left(s'|s, \pi_1(s)\right) \left[R(s, \pi_1(s), s') + \gamma U(s')\right]$$

$$U(s) = \sum_{s'} P\left(s'|s, \pi_1(s)\right) \left[10 + \gamma U(s')\right]$$

Similarly, let π_2 be policy that Liz always codes. Therefore we obtain the value function for policy π_2 as,

$$U(s) = \sum_{s'} P\left(s'|s, \pi_2(s)\right) \left[R(s, \pi_2(s), s') + \gamma U(s')\right]$$

$$U(s) = \sum_{s'} P\left(s'|s, \pi_2(s)\right) \left[4 + \gamma U(s')\right]$$

(3) Using π_1 as the current policy, and a $\gamma = 0.9$, we find the utilities of the three states as follows: Let the utilities of states SE, PIP and Unemployed be x, y, and z respectively. Therefore, using the value function that we derived for π_1 in Part (2) above, we find,

$$U(s) = \sum_{s'} P\left(s'|s, \pi_1(s)\right)\left[10 + \gamma U(s')\right]$$

$$x = 10 + 0.9 \cdot \frac{1}{4} \cdot x + 0.9 \cdot \frac{3}{4} \cdot y \tag{1}$$

$$y = 10 + 0.9 \cdot \frac{7}{8} \cdot y + 0.9 \cdot \frac{1}{8} \cdot z \tag{2}$$

$$z = 0 \tag{3}$$

Solving Equations 1, 2, and 3 simultaneously, we obtain $x \approx 53.9, y \approx 47.1$ and $z = 0$. Therefore, the utilities of the states SE, PIP, and Unemployed using the initial policy of π_1 =Netflix, are 53.9, 47.1 and 0 respectively.

Now, using the above utility values for the three states, we will update the policy using the following equation:

$$\pi \leftarrow \arg\max_{a \in A(s)} \sum_{s'} P(s'|s, a) \cdot V(s').$$

For state = SE, we find the following:

$$\arg\max_{a \in A(s)} = \begin{cases} \frac{1}{4} \cdot (10 + 53.9) + \frac{3}{4} \cdot (10 + 47.1) = 58.8 & a = \text{Netflix} \\ 1 \cdot (4 + 53.9) = 57.9 & a = \text{Code} \end{cases}$$

Therefore, the optimal action for state = SE using the above utilities is Netflix.

Similarly, for state = PIP, we find the following:

$$\arg\max_{a \in A(s)} = \begin{cases} \frac{7}{8} \cdot (10 + 47.1) + \frac{1}{8} \cdot (0) = 49.96 & a = \text{Netflix} \\ \frac{1}{4} \cdot (4 + 53.9) + \frac{3}{4} \cdot (4 + 47.1) = 52.80 & a = \text{Code} \end{cases}$$

Therefore, the optimal action for state = PIP using the above utilities is Code.

The transition probabilities from Unemployed to any other state is 0. Therefore, given the higher utility of Netflix, the optimal action for state = Unemployed is always Netflix.

Therefore the optimal policy after the first policy iteration for the states [SE, PIP, Unemployed] is [1, 0, 1] = [Netflix, Code, Netflix].

Using the above updated policy, we evaluate the policy and find the utilities of the three states as follows: Let the utilities of states SE, PIP and Unemployed be x, y, and

z respectively. Therefore, using the value function that we derived for π_1 in Part (2) above, we find,

$$U(s) = \sum_{s'} P\left(s'|s, \pi_1(s)\right)\left[10 + \gamma U(s')\right]$$

$$x = 10 + 0.9 \cdot \frac{1}{4} \cdot x + 0.9 \cdot \frac{3}{4} \cdot y \tag{4}$$

$$y = 4 + 0.9 \cdot \frac{1}{4} \cdot x + 0.9 \cdot \frac{3}{4} \cdot y \tag{5}$$

$$z = 0 \tag{6}$$

Solving Equations 4, 5, and 6 simultaneously, we obtain $x \approx 59.5, y \approx 53.5$ and $z = 0$. Therefore, the utilities of the states SE, PIP, and Unemployed using the updated policy after the 1st iteration are approximately 59.5, 53.5 and 0 respectively.

Now, using the above utility values for the three states, we will again update the policy using the following equation:

$$\pi \leftarrow \arg\max_{a \in A(s)} \sum_{s'} P(s'|s, a) \cdot V(s').$$

For state = SE, we find the following:

$$\arg\max_{a \in A(s)} = \begin{cases} \frac{1}{4} \cdot (10 + 59.5) + \frac{3}{4} \cdot (10 + .53.5) = 65.0 & a = \text{Netflix} \\ 1 \cdot (4 + 59.5) = 63.5 & a = \text{Code} \end{cases}$$

Therefore, the optimal action for state = SE using the above utilities is Netflix.

Similarly, for state = PIP, we find the following:

$$\arg\max_{a \in A(s)} = \begin{cases} \frac{7}{8} \cdot (10 + 53.5) + \frac{1}{8} \cdot (0) = 55.56 & a = \text{Netflix} \\ \frac{1}{4} \cdot (4 + 59.5) + \frac{3}{4} \cdot (4 + 53.5) = 59.0 & a = \text{Code} \end{cases}$$

Therefore, the optimal action for state = PIP using the above utilities is Code.

The transition probabilities from Unemployed to any other state is 0. Therefore, given the higher utility of Netflix, the optimal action for state = Unemployed is always Netflix.

Therefore the optimal policy after the first policy iteration for the states [SE, PIP, Unemployed] is [1, 0, 1] = [Netflix, Code, Netflix].

Since the policy has not changed in the last two iterations, we consider the policy to have converged. Therefore, the optimal policy for the states [SE, PIP, Unemployed] is [1, 0, 1] = [Netflix, Code, Netflix]. □

Please note that the action for Unemployed could be either NetFlix or Code (or None). I have used Netflix as the action for Unemployed as the default value.

(4) I implemented the policy iteration function using the given Python starter code and confirmed that my optimal policies for $\gamma = 0.9$ and $\gamma = 0.8$ are correct.

The optimal policy using $\gamma = 0.9$ for the states [SE, PIP, Unemployed] is [1, 0, 1] = [Netflix, Code, Netflix]. The action for Unemployed could be either NetFlix or Code (or None).

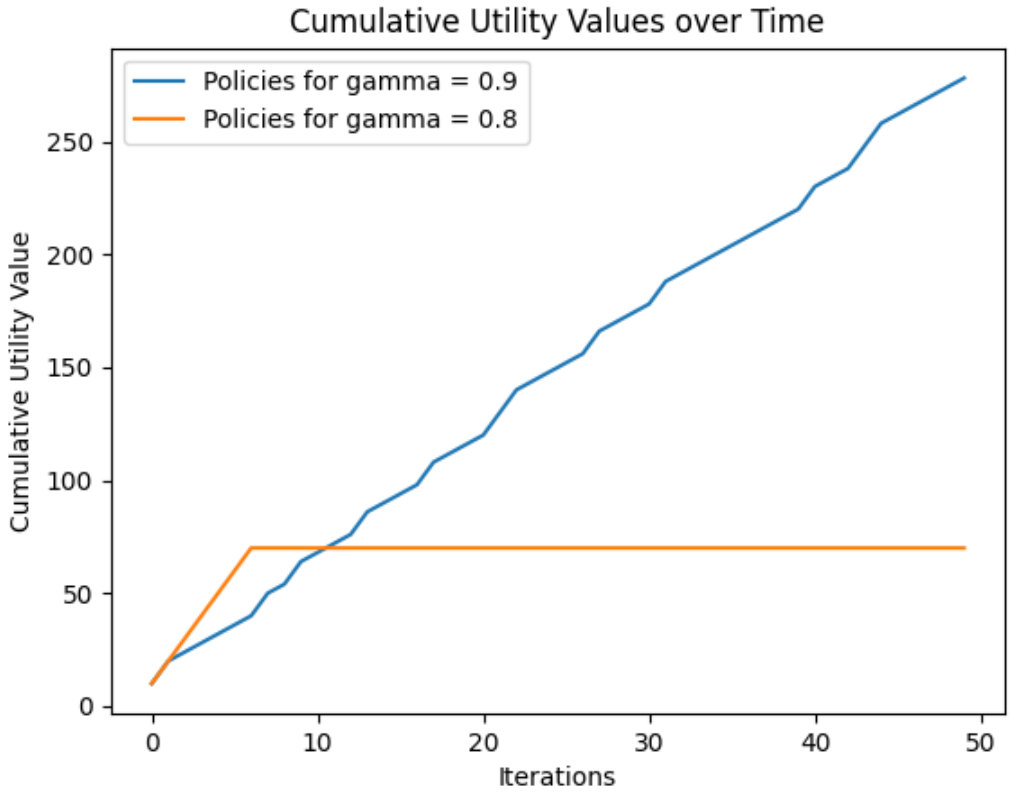
The optimal policy using $\gamma = 0.8$ for the states [SE, PIP, Unemployed] is [1, 1, 1] = [Netflix, Netflix, Netflix]. The action for Unemployed could be either NetFlix or Code (or None).

I have listed my code below, which includes a helper function called `computeUtil()`, which computes the utility of a given state, a function called `policy_evaluation()`, which computes the utilities for all the states using a given policy and finally a function called `policy_iteration()`, which iteratively improves the policy and returns the optimal policy for the given γ.

```
# Helper function
# Compute the utility of the given state (curState). For each state that
# can be reached from curState, we add the product of its utility, gamma
# and  the probability of reaching that state to the reward.
def computeUtil(curState, action, gamma, modelUtils, reward):
    util = reward[curState, action]
    for s in range(n_states):
        util += t_p[curState, action, s]*gamma*modelUtils[s]
    return util

# Policy evaluation function
def policy_evaluation(policy, modelUtils, theta, gamma, reward):
    curUtil = np.zeros(n_states,dtype=float)
    while True:
        error = 0.0
        for state in range(n_states):
            curUtil[state] = computeUtil(state, policy[state], gamma, modelUtils, re
            error = max(error, abs(curUtil[state] - modelUtils[state]))
        modelUtils = curUtil
        if error < theta*(1-gamma) / gamma:
            break
    return modelUtils
```

```
def policy_iteration(gamma):
    """
    Implements policy iteration. Returns the converged policy and a list
    of the sum of utilities at each iteration
    """
    # Utility values of the states
    modelUtils = np.zeros(n_states, dtype=float)
    theta = 1e-5 # theta determines if the change in utilities from iteration
                 # to iteration is "small enough"
    policy = np.zeros(n_states, dtype=int) # define the policy
    # Initial policy is NetFlix for all 3 states
    for i in range(len(policy)):
        policy[i] = 1

    while True:
        # Policy Evaluation
        modelUtils = policy_evaluation(policy, modelUtils, theta, gamma, r)

        # Policy Change check
        policy_stable = True

        # Policy Iteration
        for s in range(n_states):
            bestAction = None
            maxUtil = -float("inf")
            for action in range(n_actions):
                u = computeUtil(s, action, gamma, modelUtils, r)
                if u > maxUtil:
                    bestAction, maxUtil = action, u
            if maxUtil > modelUtils[s]:
                policy[s] = bestAction
                policy_stable = False
        # Determine if policy has changed between iterations
        if policy_stable:
            break
    # Return the optimal policy
    return policy
```

(5) Plot of the utility values of the two optimal policies iterating over 50 time steps is shown below for $\gamma = 0.9$ and $\gamma = 0.8$.



When the discount factor γ is closer to 1, the model favors long term reward. Therefore, it will favor a policy that avoids getting into the "Unexmployed" state, which has a reward of 0, and from which the agent (Liz) cannot reach other states. This model behavior is consistent with the plot for $\gamma = 0.9$, which yields consistently increasing cumulative utility values.

On the other hand, when the discount factor γ is reduced (for example when $\gamma = 0.8$), the agent prefers immediate reward. So, Liz doesn't mind falling into the "Unemployed" state for the near-term benefit of accumulating a higher reward by watching Netflix. However, when Liz becomes "Unemployed", she no longer accumulates any reward. This model behavior is consistent with the plot for $\gamma = 0.8$, which shows higher rewards initially than that for $\gamma = 0.9$, but shows a flat curve on the long run. The first part of this second plot has a slope of 10 as Liz is consistently receiving a reward of 10 at each time step until she becomes unemployed.

I have copied my `value_plots()` code on the next page.

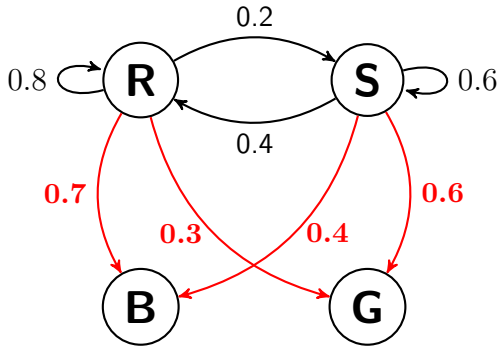
The `value_plots()` function uses a helper function called `policyUtils()`, which returns a list of cumulative values for a given policy.

```
def value_plots(policy1_vals, policy2_vals):
    """
    Plots of cumulative utility versus time for 2 different discount factors
    """
    iterations = range(0, 50)

    # Helper function to generate cumulative utilities
    # for a given policy. We use numpy.random.uniform()
    # to generate the desired probabilities
    def policyUtils(policy):
        policy_vals = []
        curState = 0
        cumUtils = 0.0
        for i in iterations:
            action = policy[curState]
            cumUtils = cumUtils + r[curState, action]
            policy_vals.append(cumUtils)
            # Probability for next action
            p = np.random.uniform(0, 1)
            if (curState == 0 and action == 0):
                curState = 0  # Continue to stay in SE
            elif (curState == 0 and action == 1):
                if (p > 0.25):  # p = 0.75
                    curState = 1 # Move to PIP
            elif (curState == 1 and action == 0):
                if (p <= 0.25):
                    curState = 0 # Move to SE
            elif (curState == 1 and action == 1):
                if (p <= 1/8):
                    curState = 2 # Move to Unemployed
        return policy_vals

    plt.plot(iterations, policyUtils(policy1), label="Policies for gamma = 0.9")
    plt.plot(iterations, policyUtils(policy2), label="Policies for gamma = 0.8")
    plt.xlabel("Iterations")
    plt.ylabel("Cumulative Utility Value")
    plt.legend()
    plt.title("Cumulative Utility Values over Time")
    plt.show()
```

PSET 3

Anusha Murali **CS182** **November 5, 2023**

Problem 3



$$T = \begin{array}{c} \\ \mathbf{R} \\ \mathbf{S} \end{array} \begin{array}{c} \begin{array}{cc} \mathbf{R} & \mathbf{S} \end{array} \\ \begin{bmatrix} 0.8 & 0.2 \\ 0.4 & 0.6 \end{bmatrix} \end{array} \qquad E = \begin{array}{c} \\ \mathbf{R} \\ \mathbf{S} \end{array} \begin{array}{c} \begin{array}{cc} \mathbf{B} & \mathbf{G} \end{array} \\ \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix} \end{array}$$

(1) Initially when Bob checked the weather channel, he found it to be rainy. Therefore, we know the posterior probabilities can be represented by the row vector $P(X_0) = [1, 0]$, where the first entry denotes the probability of a rainy day and the second entry denotes the probability of a sunny day. We use X and E for the hidden variable (weather state) and evidence of observed variable (Alice's mood) respectively. From the problem statement, we find the transition probabilities of the weather states can be represented by matrix T and the emission probabilities can be represented by matrix E as shown above.

Day 1:

The prediction from $t = 0$ to $t = 1$ is,

$$P(X_1) = P(X_0) \cdot \mathrm{T} = [1, 0] \begin{bmatrix} 0.8 & 0.2 \\ 0.4 & 0.6 \end{bmatrix} = \langle 0.8, 0.2 \rangle$$

Since Alice's mood on the 1st day was good (G), $P(e_1|X_1) = \langle 0.3, 0.6 \rangle$. Now, in the following update step, using Bayes Theorem, we multiply $P(X_1)$ by the probability of the evidence for $t = 1$ and normalize:

$$\begin{aligned} P(X_1|e_1) &= \alpha P(e_1|X_1) P(X_1) = \alpha \langle 0.3, 0.6 \rangle \langle 0.8, 0.2 \rangle \\ &= \alpha \langle 0.24, 0.12 \rangle \\ &\approx \langle 0.667, 0.333 \rangle \end{aligned}$$

Day 2:

The prediction from $t = 1$ to $t = 2$ is,

$$\begin{aligned} P(X_2|e_1) &= \sum_{x_1} P(X_2|x_1) P(x_1|e_1) \\ &= P(X_1|e_1) \cdot \mathrm{T} = \langle 0.667, 0.333 \rangle \begin{bmatrix} 0.8 & 0.2 \\ 0.4 & 0.6 \end{bmatrix} \\ &= \langle 2/3, 1/3 \rangle \end{aligned}$$

Since Alice's mood on the 2nd day was good (G), $P(e_2|X_2) = \langle 0.3, 0.6\rangle$ Now, in the following update step, we multiply $P(X_2|e_1)$ by the probability of the evidence for $t = 2$ and normalize:

$$\begin{aligned} P(X_2|e_1, e_2) &= \alpha P(e_2|X_2)P(X_2|e_1) = \alpha\langle 0.3, 0.6\rangle\langle 2/3, 1/3\rangle \\ &= \alpha\langle 0.2, 0.2\rangle \\ &= \langle 0.5, 0.5\rangle \end{aligned}$$

Day 3:

The prediction from $t = 2$ to $t = 3$ is,

$$\begin{aligned} P(X_3|e_1, e_2) &= \sum_{x_2} P(X_3|e_2)P(x_2|e_1, e_2) \\ &= P(X_2|e_1, e_2) \cdot \mathrm{T} = \langle 0.5, 0.5\rangle \begin{bmatrix} 0.8 & 0.2 \\ 0.4 & 0.6 \end{bmatrix} \\ &= \langle 0.6, 0.4\rangle \end{aligned}$$

Since Alice's mood on the 3rd day was bad (B), $P(e_3|X_3) = \langle 0.7, 0.4\rangle$. Now, in the following update step, we multiply $P(X_3|e_1, e_2)$ by the probability of the evidence for $t = 3$ and normalize:

$$\begin{aligned} P(X_3|e_1, e_2, e_3) &= \alpha P(e_3|X_3)P(X_3|e_1, e_2) = \alpha\langle 0.7, 0.4\rangle\langle 0.6, 0.4\rangle \\ &= \alpha\langle 0.42, 0.16\rangle \\ &= \langle 0.7241, 0.2759\rangle \end{aligned}$$

Day 4:

The prediction from $t = 3$ to $t = 4$ is,

$$\begin{aligned} P(X_4|e_1, e_2, e_3) &= \sum_{x_3} P(X_4|e_3)P(x_3|e_1, e_2, e_3) \\ &= P(X_3|e_1, e_2, e_3) \cdot \mathrm{T} = \langle 0.7241, 0.2759\rangle \begin{bmatrix} 0.8 & 0.2 \\ 0.4 & 0.6 \end{bmatrix} \\ &\approx [0.68964, 0.31036] \end{aligned}$$

Therefore, the probability that it will be rainy on day 4 is 0.68964. □

(2) **Method I: Viterbi Algorithm** We use the Viterbi algorithm, which is linear in the number of days, to determine the most likely sequence of weather states below.

Day 1

Initially when Bob checked the weather channel, he found it to be rainy. We can compute the probabilities of Rainy versus Sunny on Day 1 as follows:

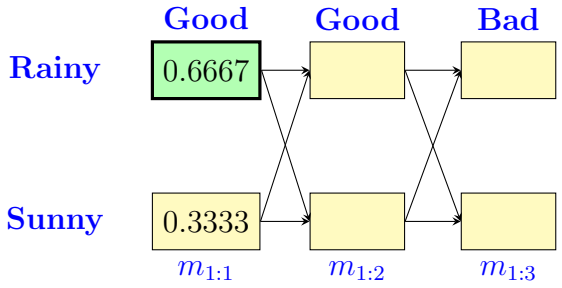
The prediction from $t = 0$ to $t = 1$ is,

$$P(X_1) = P(X_0) \cdot \mathrm{T} = [1, 0] \begin{bmatrix} 0.8 & 0.2 \\ 0.4 & 0.6 \end{bmatrix} = \langle 0.8, 0.2 \rangle$$

Since Alice's mood on the 1st day was good (G), $P(e_1|X_1) = \langle 0.3, 0.6 \rangle$. Now, in the following update step, using Bayes Theorem, we multiply $P(X_1)$ by the probability of the evidence for $t = 1$ and normalize:

$$\begin{aligned} P(X_1|e_1) &= \alpha P(e_1|X_1)P(X_1) = \alpha\langle 0.3, 0.6\rangle\langle 0.8, 0.2\rangle \\ &= \alpha\langle 0.24, 0.12\rangle \\ &\approx \langle 0.667, 0.333\rangle \end{aligned}$$

Hence, we fill the Rainy and Sunny boxes for Day 1 with their corresponding probabilities as follows:

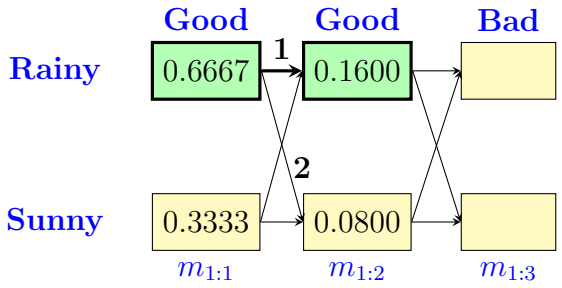


Using the Viterbi algorithm, we select the box with the larger probability of 0.6667, which corresponds to Rainy. This box is shown in green above.

Day 2

Alice's mood on the 2nd day was Good. Hence the probability of the 2nd day being Rainy and Alice's mood being Good is $0.6667 \times 0.8 \times 0.3 \approx 0.1600$ (labeled 1 in diagram), and the the probability of the 2nd day being Sunny and Alice's mood being Good is $0.6667 \times 0.2 \times 0.6 \approx 0.0800$ (labeled 2 in diagram).

Hence, we fill the Rainy and Sunny boxes for Day 2 with their corresponding probabilities as follows:

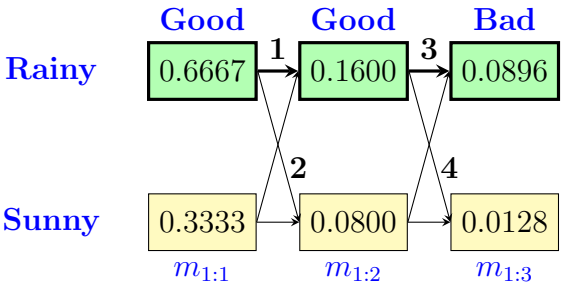


Using the Viterbi algorithm, we select the box with the larger probability of 0.1600, which corresponds to Rainy. This box is shown in green above.

Day 3

Alice's mood on the 3rd day was Bad. Hence the probability of the 3rd day being Rainy and Alice's mood being Bad is $0.1600 \times 0.8 \times 0.7 \approx 0.0896$ (labeled 3 in diagram), and the the probability of the 3rd day being Sunny and Alice's mood being Bad is $0.1600 \times 0.2 \times 0.4 \approx 0.0128$ (labeled 4 in diagram).

Hence, we fill the Rainy and Sunny boxes for Day 3 with their corresponding probabilities as follows:



Using the Viterbi algorithm, we select the box with the larger probability of 0.0896, which corresponds to Rainy. This box is shown in green above.

The green boxes in the above diagram corresponds to the mostly likely weather states over the three days. Hence, under Bob's model, the most likely sequence of weather states over the three days is Rainy, Rainy, Rainy. □

Method II: Enumeration Alternatively, we can find the most likely sequence of weather states by computing the likely probabilities over all the possible sequences. If we denote the weather states as $X = X_1, X_2, X_3$, and Alice's moods as $Y = Y_1, Y_2, Y_3$, we are interested in finding out the sequence X, which maximizes the probability of Alice's moods that were observed over the three days. In other words, we are interested in computing the following:

$$\underset{X=X_1,X_2,X_3}{\text{argmax}} \; P(X = X_1, X_2, X_3 | Y = Y_1, Y_2, Y_3) \propto \underset{X=X_1,X_2,X_3}{\text{argmax}} \prod P(X_i|X_{i-1})P(Y_i|X_i)$$

The right most expression results from the Bayes' theorem.

A day could be either rainy (R) or sunny (S). Hence there are 2 possible weather states each day, there are $2^3 = 8$ possible weather sequences over the three days. Hence the possible hidden state sequences are namely, RRR, RRS, RSR, RSS, SRR, SRS, SSR, and SSS. The observed states are Alice's mood over the three days, namely Good (G), Good (G) and Bad (B).

Using the given transmission probabilities and emission probabilities, we will evaluate the quantity $\prod P(X_i|X_{i-1})P(Y_i|X_i)$ for each of the above 8 weather state sequences below. Since Bob knew it rained the day before, the probability of raining on the 1st day is $X_1 = 0.8$ and the probability of being sunny on the 1st day is $X_1 = 0.2$.

Weather sequence	$\prod P(X_i\|X_{i-1})P(Y_i\|X_i)$	Probability
RRR	$P(R)P(G\|R) \times P(R\|R)P(G\|R) \times P(R\|R)P(B\|R)$ $= (0.8 \times 0.3)(0.8 \times 0.3)(0.8 \times 0.7)$	0.032256
RRS	$P(R)P(G\|R) \times P(R\|R)P(G\|R) \times P(S\|R)P(B\|S)$ $= (0.8 \times 0.3)(0.8 \times 0.3)(0.2 \times 0.4)$	0.004608
RSR	$P(R)P(G\|R) \times P(S\|R)P(G\|S) \times P(R\|S)P(B\|R)$ $= (0.8 \times 0.3)(0.2 \times 0.6)(0.4 \times 0.7)$	0.008064
RSS	$P(R)P(G\|R) \times P(S\|R)P(G\|S) \times P(S\|S)P(B\|S)$ $= (0.8 \times 0.3)(0.2 \times 0.6)(0.6 \times 0.4)$	0.006912
SRR	$P(S)P(G\|S) \times P(R\|S)P(G\|R) \times P(R\|R)P(B\|R)$ $= (0.2 \times 0.6)(0.4 \times 0.3)(0.8 \times 0.7)$	0.008064
SRS	$P(S)P(G\|S) \times P(R\|S)P(G\|R) \times P(S\|R)P(B\|S)$ $= (0.2 \times 0.6)(0.4 \times 0.3)(0.2 \times 0.4)$	0.001152
SSR	$P(S)P(G\|S) \times P(S\|S)P(G\|S) \times P(R\|S)P(B\|R)$ $= (0.2 \times 0.6)(0.6 \times 0.6)(0.4 \times 0.7)$	0.012096
SSS	$P(S)P(G\|S) \times P(S\|S)P(G\|S) \times P(S\|S)P(B\|S)$ $= (0.2 \times 0.6)(0.6 \times 0.6)(0.6 \times 0.4)$	0.010368

Therefore, we find that $\text{argmax}_{X=X_1,X_2,X_3} \prod P(X_i|X_{i-1})P(Y_i|X_i) = RRR$, which occurs with the maximum probability of 0.032256 over the all 8 possible weather sequences.

Hence, under Bob's model, the most likely sequence of weather states over the three days is Rainy, Rainy, Rainy. □

PSET 3

Anusha Murali **CS182** **November 5, 2023**

Problem 4

(1) Let $P = (x, y)$ be a location on the grid, and $h(P)$ denote the expected time taken to get to the subway station from P based on this open-loop strategy. Let a be the action taken by Evan at P, and let $a(P)$ denote the location of Evan after taking the action a at P. Note that we should be having another parameter to define our set of available actions, which is the current state of traffic lights, but we omit that for brevity.

We know that $h(2, 2) = 0$ because Evan is already at the station. Now, if Evan first arrives at $(2, 1)$, the only choice is to go North, so his movement is independent of the East-West signal. Therefore, his expected time to reach the subway station is the following.

$$h(2,1) = \sum_{i=0}^{\infty} 0.4 \times 0.6^i (1+i) = \frac{5}{2}$$

Note that we could have obtained the above expected wait time by recognizing that the number of color changes is a geometric distribution where the success (green) has a probability of $p = 0.4$ and failure (red) has a probability of $q = 0.6$. Therefore the expected wait time for the first green light is $\frac{q}{p} = \frac{0.6}{0.4} = 1.5$ minutes. Since it takes Evan 1 minute to walk a block, we find $h(2, 1) = 1.5 + 1 = 2.5$ minutes.

Similarly, if Evan first arrives at $(1, 2)$, the only choice is to go East, so his movement is independent of the North-South signal. Therefore, his expected time to reach the subway station is the following.

$$h(1,2) = \sum_{i=0}^{\infty} 0.3 \times 0.7^i (1+i) = \frac{10}{3}$$

If Evan first arrives at $(2, 0)$, the only choice is to go North, so his movement is independent of the East-West signal. Therefore, his expected time to reach the station is the following.

$$h(2,0) = \sum_{i=0}^{\infty} 0.4 \times 0.6^i (1+i) + h(2,1) = 5$$

Similarly, if Evan first arrives at $(0, 2)$, the only choice is to go East, so his movement is independent of the North-South signal. Therefore, his expected time to reach the station is the following.

$$h(0,2) = \sum_{i=0}^{\infty} 0.3 \times 0.7^i (1+i) + h(1,2) = \frac{20}{3}$$

Next, we compute $h(1,1)$ as follows:

$$h(1,1) = 0.6 \times 0.7 \times (1 + h(1,1)) + 0.3 \times (1 + h(2,1)) + 0.4 \times 0.7 \times (1 + h(1,2)),$$

where the first term on the RHS is the sum of the expected wait time at (1,1) due to both stoplights being red and the block travel time, the second term on the RHS is the sum of the expected wait time for the East light to become green and the block travel time, and the third term on the RHS is the sum of the expected wait time for the North light to become green while the East light is being red and the block travel time.

Simplifying and solving, we find $h(1,1) = \frac{805}{174}$.

Next, we compute $h(1,0)$ as follows:

$$h(1,0) = 0.6 \times 0.7 \times (1 + h(1,0)) + 0.3 \times (1 + h(2,0)) + 0.4 \times 0.7 \times (1 + h(1,1))$$

Simplifying and solving, we find $h(1,0) = \frac{16510}{2523}$.

Next, we compute $h(0,1)$ as follows:

$$h(0,1) = 0.6 \times 0.7 \times (1 + h(0,1)) + 0.3 \times (1 + h(1,1)) + 0.4 \times 0.7 \times (1 + h(0,2))$$

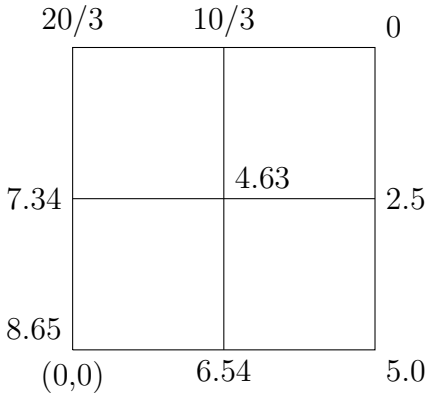
Simplifying and solving, we find $h(0,1) = \frac{37015}{5046}$.

Finally, we compute $h(0,0)$ as follows:

$$h(0,0) = 0.6 \times 0.7 \times (1 + h(0,0)) + 0.3 \times (1 + h(1,0)) + 0.4 \times 0.7 \times (1 + h(0,1))$$

Simplifying and solving, we find $h(0,0) = \frac{632905}{73167} \approx 8.65$.

The following diagram shows the "cost to go" or the expected travel time to reach the subway station from all the intersections. Therefore, the open-loop policy gives us an expected travel time of ≈ 8.65 minutes.



(2) We compute the expected time for Evan's travel using the closed-loop policy as follows. Let $P = (x, y)$ be a location on the grid, and $f(P)$ denote the **expected time** taken to get to the station from P based on this closed-loop strategy. Let a be the action taken by Evan at P, and let $a(P)$ denote the location of Evan after taking the action a at P. Note that we should be having another parameter to define our set of available actions, which is the current state of traffic lights, but we omit that for brevity.

Our strategy is as follows. Let A_P be the set of valid actions that can be taken by Evan at point P ($A_P \subseteq \{\textsf{East}, \textsf{North}, \textsf{Stay}\}$). In other words, the valid actions could be a subset of moving east, moving north, or not moving at all – when both traffic lights say Red, then Stay is the only valid action Evan can take. When only the North-South light is Green, then Evan can only take North, while when only the East-West light says Green, then Evan can only take East, otherwise Evan can take either North or East. We do not allow Evan to stay at a particular location when at least one of the lights says Green.

Formally from the Bellman's Equation we can say that for each P, the expected wait time is

$$f_k(P) = \arg\min_{a \in A_P} (1 + f_{k+1}(P)) .$$

Hence the optimal action a_P^* will be added to the optimal policy, π^*, where

$$a_P^* \in \arg\min_{a \in A_P} \{1 + f(a(P))\}.$$

Using the Principle of Optimality, an optimal policy for our problem can be subdivided into two components:

- An optimal first action a_P^*, where $P = (2, 2)$.
- Followed by an optimal policy from the successor state P', where $P' = (2, 1)$ or $P' = (1, 2)$.

In our case, $f((2, 2)) = 0$. The optimal strategy at $(2, 1)$ is to move North as soon as the North-South signal says Green, and the optimal strategy at $(1, 2)$ is to move East as soon as the East-West signal says Green. This implies that, similar to Part 1, $f((2, 1)) = 5/2$ and $f((1, 2)) = 10/3$.

As an example for our recursion, we show what Evan would do at position $(1, 1)$. When both the traffic signals say Red, he just stays. When only one direction says Green, he just goes in that direction. When both lights say Green, he goes East because the expected wait time at $(2, 1)$ is lower.

Our memoization strategy will store the values of $f(P)$ for all the valid P in the grid. From Part 1, we have the following.

$$f(2, 2) = 0, \quad f(2, 1) = \frac{5}{2}, \quad f(1, 2) = \frac{10}{3}, \quad f(2, 0) = 5, \quad f(0, 2) = \frac{20}{3}$$

We can fill the rest of the grid by solving recursive equations as follows.

First, we compute $f(1,1)$:

$$\begin{aligned} f(1,1) = {} & 0.6 \times 0.7 \times (1 + f(1,1)) + 0.4 \times 0.7 \times (1 + f(1,2)) \\ & + 0.6 \times 0.3 \times (1 + f(2,1)) \\ & + 0.4 \times 0.3 \times (1 + \min\{f(2,1), f(1,2)\}) \end{aligned}$$

Simplifying and solving, we find $f(1,1) = \dfrac{805}{174}$.

Next, we compute $f(1,0)$:

$$\begin{aligned} f(1,0) = {} & 1 + 0.42 \times f(1,0) + 0.28 \times f(1,1) \\ & + 0.18 \times f(2,0) \\ & + 0.12 \times \min\{f(1,1), f(2,0)\} \end{aligned}$$

Simplifying and solving, we find $f(1,0) = \dfrac{16315}{2523}$.

Next, we compute $f(0,1)$:

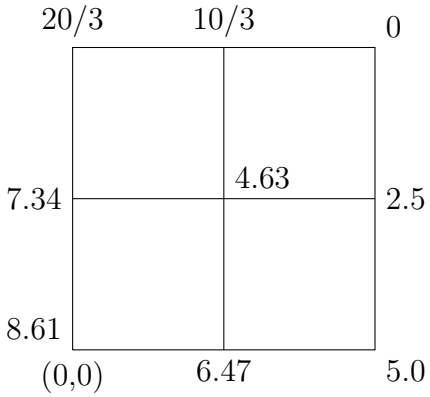
$$\begin{aligned} f(0,1) = {} & 1 + 0.42 \times f(0,1) + 0.28 \times f(0,2) \\ & + 0.18 \times f(1,1) \\ & + 0.12 \times \min\{f(0,2), f(1,1)\} \end{aligned}$$

Simplifying and solving, we find $f(0,1) = \dfrac{37015}{5046}$.

Finally, we compute $f(0,0)$:

$$\begin{aligned} f(0,0) = {} & 1 + 0.42 \times f(0,0) + 0.28 \times f(0,1) \\ & + 0.18 \times f(1,0) + 0.12 \times \min\{f(0,1), f(1,0)\} \end{aligned}$$

Simplifying and solving, we find $f(0,0) = \dfrac{629980}{73167} \approx 8.61$. The following diagram shows the "cost to go" or the expected travel time to reach the subway station from all the intersections.



Using the above expected times to arrive at (2,2) from each intersections, we find the optimal path under the closed-loop policy for any arbitrary arrival time at (0,0). Due to the stochastic nature of the stoplights, the optimal policy, which results in the least expected travel to Evan, is established depending on Evan's arrival time at (0, 0).

Therefore, this strategy gives us a time of approximately 8.61 minutes.

So, the value of information, which is the difference between the expected travel time between the open-loop and the closed-loop policies, is approximately $8.65 - 8.61 = 0.04$.

PSET 3

Anusha Murali **CS182** **November 5, 2023**

Problem 5

(1) I worked on this by myself. I did not use any other resources besides the lecture slides and the textbook.

(2) I spent 35 hours (5 on the coding part, and 30 on the theory part) on this assignment.