CS 221: Artificial Intelligence: Principles and Techniques

Assignment 4: Peeking Blackjack

May 1, 2022

SUNet ID: jchan7
Name: Jason Chan
Collaborators: None

By turning in this assignment, I agree by the Stanford honor code and declare that all of this is my own work.

Before you get started, please read the Assignments section on the course website thoroughly.

The search algorithms explored in the previous assignment work great when you know exactly the results of your actions. Unfortunately, the real world is not so predictable. One of the key aspects of an effective AI is the ability to reason in the face of uncertainty.

Markov decision processes (MDPs) can be used to formalize uncertain situations. In this homework, you will implement algorithms to find the optimal policy in these situations. You will then formalize a modified version of Blackjack as an MDP, and apply your algorithm to find the optimal policy. Finally, you will explore using MDPs for modeling a real-world scenario, specifically rising sea levels.

Problem 1: Value Iteration

In this problem, you will perform the value iteration updates manually on a very basic game just to solidify your intuitions about solving MDPs. The set of possible states in this game is $\mathcal{S} = \{-2, -1, 0, +1, +2\}$ and the set of possible actions is $\mathcal{A} = \{a_1, a_2\}$. The initial state is 0 and there are two terminal states, -2 and $+2$. Recall that the transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ encodes the probability of transitioning to a next state s' after being in state s and taking action a as $\mathcal{T}(s'|s, a)$. In this MDP, the transition dynamics are given as follows:

$\forall i \in \{-1, 0, 1\} \subset \mathcal{S}$,

- $\mathcal{T}(i-1|i, a_1) = 0.8$ and $\mathcal{T}(i+1|i, a_1) = 0.2$
- $\mathcal{T}(i-1|i, a_2) = 0.7$ and $\mathcal{T}(i+1|i, a_2) = 0.3$

Think of this MDP as a chain formed by states $\{-2, -1, 0, +1, +2\}$. In words, action a_1 has a 80% chance of moving the agent backwards in the chain and a 20% chance of moving the agent forward. Similarly, action a_2 has a 70% of sending the agent backwards and a 30% chance of moving the agent forward. We will use a discount factor $\gamma = 1$.

The reward function for this MDP is $\mathcal{R}(s, a, s') = \begin{cases} 10 & s' = -2 \\ 50 & s' = +2 \\ -5 & \text{otherwise} \end{cases}$

a. [3 points] What is the value of $V_i^\star(s)$ for each state in $\mathcal{S}$ after each iteration $i = \{0, 1, 2\}$ of Value Iteration? Please write down the values from all iterations. Recall that $\forall s \in \mathcal{S}$, $V_0^\star(s) = 0$ and, for any terminal state s_{terminal}, $V^\star(s_{\text{terminal}}) = 0$. In other words, all values are 0 at iteration 0 and terminate states always have a value of 0.

What we expect: The $V_i^\star(s)$ of all 5 states after each iteration. In total, 15 values should be reported.

Your Solution:

state (s)	$V_0{}^*(s)$	$V_1{}^*(s)$	$V_2{}^*(s)$
-2	0	0	0
-1	0	7	6
0	0	-5	3.35
+1	0	11.5	8
+2	0	0	0

b. [1 point] Using $V_2^\star(\cdot)$, what is the corresponding optimal policy $\pi^\star$ for all non-terminal states?

What we expect: A few state action pairs to express the optimal policy.

Your Solution:

state (s)	$\pi^*(s)$
-1	a_1
0	a_2
+1	a_2

Problem 2: Transforming MDPs

Equipped with an understanding of a basic algorithm for computing optimal value functions in MDPs, let's gain intuition about the dynamics of MDPs which either carry some special structure, or are defined with respect to a different MDP.

a. [4 points] Suppose we have an MDP with states States and a discount factor $\gamma < 1$, but we have an MDP solver that can only solve MDPs with discount factor of 1. How can we leverage the MDP solver to solve the original MDP?

Let us define a new MDP with states $\text{States}' = \text{States} \cup \{o\}$, where o is a new state. Let's use the same actions $\text{Actions}'(s) = \text{Actions}(s)$ but we need to keep the discount $\gamma' = 1$. Your job is to define new transition probabilities $T'(s, a, s')$ and rewards $\text{Reward}'(s, a, s')$ in terms of the old MDP such that the optimal values $V_{\text{opt}}(s)$ for all $s \in \text{States}$ are equal under the original MDP and the new MDP.

HINT: If you're not sure how to approach this problem, go back to Tatsu's notes from the first MDP lecture and read closely the slides on convergence, toward the end of the deck.

What we expect: A few transition probabilities and reward functions written in mathematical expressions, followed by a short proof to show that the two optimal values are equal. Try to use the same symbols as the question.

Your Solution: Tatsu's notes inform us that we can *reinterpret* $\gamma < 1$ *as introducing a new transition state to a special end state with probability* $(1 - \gamma)$*, multiplying all other transitions by* γ *and setting the discount to 1.* Let's rewrite the optimal values and Q-values for the reinterpretation of $\gamma < 1$:

Suppose we define **Option A**

$$T'(s, a, s') = \begin{cases} 1 - \gamma & \text{if } s' \text{ special end state} \\ \gamma T(s, a, s) & \text{otherwise} \end{cases} \tag{1}$$

$$R'(s, a, s') = \begin{cases} 0 & \text{if } s' \text{ special end state} \\ \frac{R(s,a,s')}{\gamma} & \text{otherwise} \end{cases} \tag{2}$$

Then Q_{Opt} for $\gamma < 1$ becomes the below since we add the expected value of the special end state and apply the reinterpretation and the above definitions

$$Q_{Opt} = \gamma \sum_{s'} T'(s, a, s')[\frac{R'(s, a, s')}{\gamma} + (1)V'_{Opt}(s')] + (1 - \gamma)0 \tag{3}$$

Which takes us back to the below. So we can apply our current MDP solver which only works for $\gamma = 1$ and apply if for cases where $\gamma < 1$ if we also use the definitions in option A.

$$Q_{Opt} = \sum_{s'} T'(s,a,s')[R'(s,a,s') + \gamma V'_{Opt}(s')] \tag{4}$$

Suppose we define **Option B**

$$T'(s,a,s') = \begin{cases} 1-\gamma & \text{if } s' \text{ special end state} \\ \gamma T(s,a,s') & \text{otherwise} \end{cases} \tag{5}$$

$$R'(s,a,s') = \begin{cases} \sum_{s'} T(s,a,s')R(s,a,s') & \text{if } s' \text{ special end state} \\ R(s,a,s') & \text{otherwise} \end{cases} \tag{6}$$

Then we see that Q_{Opt} for $\gamma < 1$ becomes the below when we substitute in the definitions in option B and apply Tatsu's reinterpretation.

$$Q_{Opt} = \gamma \sum_{s'} T'(s,a,s')[R(s,a,s') + (1)V'_{Opt}(s')] + (1-\gamma)(\sum_{s'} T(s,a,s')R(s,a,s')) \tag{7}$$

$$Q_{Opt} = \gamma \sum_{s'} T'(s,a,s')(R(s,a,s')) + \gamma \sum_{s'} T'(s,a,s')(1)V'_{Opt}(s') + (1-\gamma)(\sum_{s'} T(s,a,s')R(s,a,s')) \tag{8}$$

$$Q_{Opt} = (1) \sum_{s'} T'(s,a,s')(R(s,a,s')) + \gamma \sum_{s'} T'(s,a,s')(1)V'_{Opt}(s') \tag{9}$$

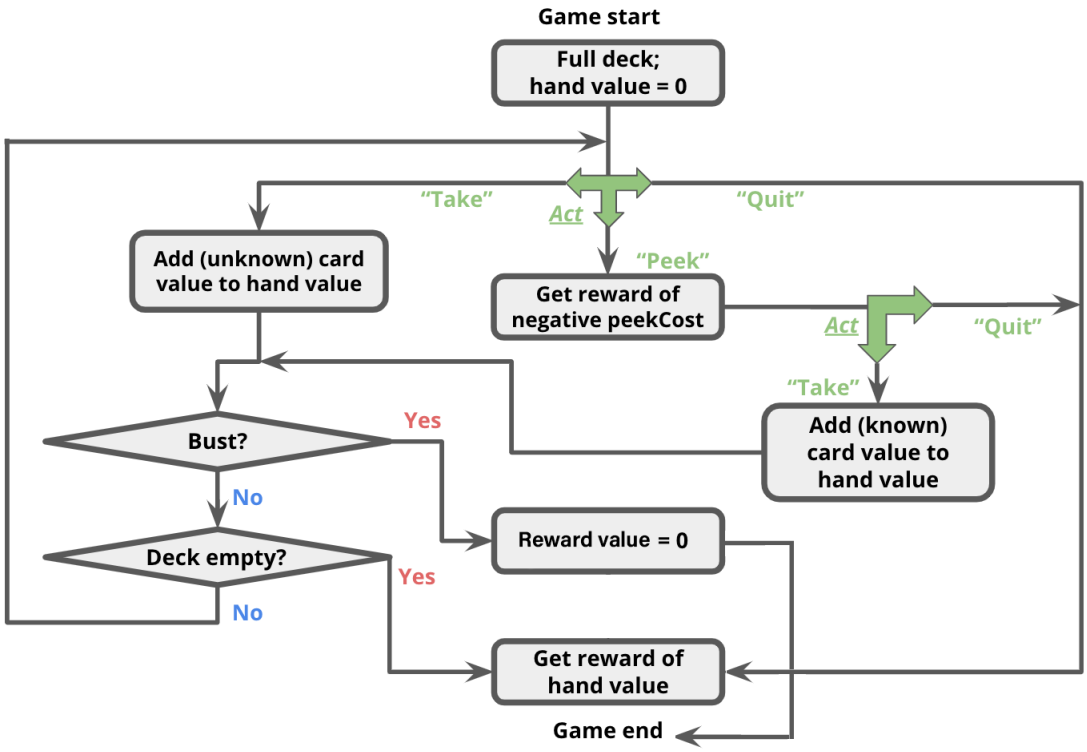
$$Q_{Opt} = \sum_{s'} T'(s,a,s')[(R(s,a,s')) + \gamma V'_{Opt}(s')] \tag{10}$$

Which takes us back to the familiar form. So we can apply our current MDP solver which only works for $\gamma = 1$ and apply if for cases where $\gamma < 1$ if we also use the definitions in option B.

Problem 3: Peeking Blackjack

Now that we have gotten a bit of practice with general-purpose MDP algorithms, let's use them to play (a modified version of) Blackjack. For this problem, you will be creating an MDP to describe states, actions, and rewards in this game. More specifically, after reading through the description of the state representation and actions of our Blackjack game below, you will implement the transition and reward function of the Blackjack MDP inside `succAndProbReward()`.

For our version of Blackjack, the deck can contain an arbitrary collection of cards with different face values. At the start of the game, the deck contains the same number of each cards of each face value; we call this number the 'multiplicity'. For example, a standard deck of 52 cards would have face values $[1, 2, \ldots, 13]$ and multiplicity 4. You could also have a deck with face values $[1, 5, 20]$; if we used multiplicity 10 in this case, there would be 30 cards in total (10 each of 1s, 5s, and 20s). The deck is shuffled, meaning that each permutation of the cards is equally likely.



The game occurs in a sequence of rounds.

In each round, the player has three actions available to her:

- a_{take} - Take the next card from the top of the deck.
- a_{peek} - Peek at the next card on the top of the deck.
- a_{quit} - Stop taking any more cards.

In this problem, your state s will be represented as a 3-element tuple:

```
(totalCardValueInHand, nextCardIndexIfPeeked, deckCardCounts)
```

As an example, assume the deck has card values $[1, 2, 3]$ with multiplicity 2, and the threshold is 4. Initially, the player has no cards, so her total is 0; this corresponds to state `(0, None, (2, 2, 2))`.

- For a_{take}, the three possible successor states (each with equal probability of 1/3) are:

  ```
  (1, None, (1, 2, 2))
  (2, None, (2, 1, 2))
  (3, None, (2, 2, 1))
  ```

  Three successor states have equal probabilities because each face value had the same amount of cards in the deck. In other words, a random card that is available in the deck is drawn and its corresponding count in the deck is then decremented. Remember that `succAndProbReward()` will expect you return all three of the successor states shown above. Note that $\mathcal{R}(s, a_{\text{take}}, s') = 0, \forall s, s' \in \mathcal{S}$. Even though the agent now has a card in her hand for which she may receive a reward at the end of the game, the reward is not actually granted until the game ends (see termination conditions below).

- For a_{peek}, the three possible successor states are:

  ```
  (0, 0, (2, 2, 2))
  (0, 1, (2, 2, 2))
  (0, 2, (2, 2, 2))
  ```

  Note that it is not possible to peek twice in a row; if the player peeks twice in a row, then `succAndProbReward()` should return `[]`. Additionally, $\mathcal{R}(s, a_{\text{peek}}, s') = -\text{peekCost}, \forall s, s' \in \mathcal{S}$. That is, the agent will receive an immediate reward of `-peekCost` for reaching any of these states.
  Things to remember about the states after taking a_{peek}:

  - From `(0, 0, (2, 2, 2))`, taking a card will lead to the state `(1, None, (1, 2, 2))` deterministically (that is, with probability 1.0).
  - The second element of the state tuple is not the face value of the card that will be drawn next, but the index into the deck (the third element of the state tuple) of the card that will be drawn next. In other words, the second element will always be between 0 and `len(deckCardCounts)-1`, inclusive.

- For a_{quit}, the resulting state will be `(0, None, None)`. (Remember that setting the deck to `None` signifies the end of the game.)

The game continues until one of the following termination conditions becomes true:

- The player chooses a_{quit}, in which case her reward is the sum of the face values of the cards in her hand.
- The player chooses a_{take} and "goes bust". This means that the sum of the face values of the cards in her hand is strictly greater than the threshold specified at the start of the game. If this happens, her reward is 0.
- The deck runs out of cards, in which case it is as if she selects a_{quit}, and she gets a reward which is the sum of the cards in her hand. *Make sure that if you take the last card and go bust, then the reward becomes 0 not the sum of values of cards.*

As another example with our deck of $[1, 2, 3]$ and multiplicity 1, let's say the player's current state is `(3, None, (1, 1, 0))`, and the threshold remains 4.

- For a_{quit}, the successor state will be `(3, None, None)`.
- For a_{take}, the successor states are `(3 + 1, None, (0, 1, 0))` or `(3 + 2, None, None)`. Each has a probability of 1/2 since 2 cards remain in the deck. Note that in the second successor state, the deck is set to `None` to signify the game ended with a bust. You should also set the deck to `None` if the deck runs out of cards.

a. [15 points] Implement the game of Blackjack as an MDP by filling out the `succAndProbReward()` function of class `BlackjackMDP`.

Problem 4: Learning to Play Blackjack

So far, we've seen how MDP algorithms can take an MDP which describes the full dynamics of the game and return an optimal policy. But suppose you go into a casino, and no one tells you the rewards or the transitions. We will see how reinforcement learning can allow you to play the game and learn its rules & strategy at the same time!

a. [8 points] You will first implement a generic Q-learning algorithm `QLearningAlgorithm`, which is an instance of an `RLAlgorithm`. As discussed in class, reinforcement learning algorithms are capable of executing a policy while simultaneously improving that policy. Look in `simulate()`, in `util.py` to see how the `RLAlgorithm` will be used. In short, your `QLearningAlgorithm` will be run in a simulation of the MDP, and will alternately be asked for an action to perform in a given state (`QLearningAlgorithm.getAction`), and then be informed of the result of that action (`QLearningAlgorithm.incorporateFeedback`), so that it may learn better actions to perform in the future.

   We are using Q-learning with function approximation, which means $\hat{Q}^\star(s, a) = \mathbf{w} \cdot \phi(s, a)$, where in code, $\mathbf{w}$ is `self.weights`, ϕ is the `featureExtractor` function, and $\hat{Q}^\star$ is `self.getQ`.

   We have implemented `QLearningAlgorithm.getAction` as a simple ϵ-greedy policy. Your job is to implement `QLearningAlgorithm.incorporateFeedback()`, which should

take an (s, a, r, s') tuple and update `self.weights` according to the standard Q-learning update.

b. [2 points] This function simulates using your Q-learning code and the `identityFeatureExtractor()` on 1) a small MDP and 2) a large MDP, each with 30000 trials. For the small MDP, how does the Q-learning policy compare with a policy learned by value iteration (i.e., for how many states do they produce a different action)? We have provided a function in the grader `4b-helper` which does this for you.

Now, for the large MDP, how does the policy learned in this case compare to the policy learned by value iteration? What went wrong?

What we expect: A sentence explaining whether Q-learning is better or worse than ValueIteration on the small MDP. A sentence explaining whether Q-learning is better or worse than ValueIteration on the large MDP. 2-3 sentences describing why you think this behavior occurred.

Your Solution: Results from 4b-helper function comparing the difference between QLearning vs. Value Iteration:

- Small MDP: 2 differences out of 27 states (7.4 per cent)
- Large MDP: 866 differences out of 2745 total states (31.5 per cent)

In general, Value Iteration is better than QLearning because Value Iteration actually finds the objective optimal policy ('truth') for a given MDP. QLearning can only ever compute the estimated optimal policy so it will always be equal or worse than Value Iteration.

For a smaller MDP, QLearning performs reasonably well because the number of states is small.

For a larger MDP, QLearning performs significantly worse than Value Iteration. With so many states, it's much harder for QLearning to evaluate the QValues for each state action pair. The other reason that QLearning didn't perform as well is that the current implementation in 4b exploits states rather than explore (ql.explorationProb = 0.0), which would improve state exploration when the state space is large.

c. [5 points] To address the problems explored in the previous exercise, let's incorporate some domain knowledge to improve generalization. This way, the algorithm can use what it has learned about some states to improve its prediction performance on other states. Implement `blackjackFeatureExtractor` as described in the code comments. Using this feature extractor, you should be able to get pretty close to the optimum values on the `largeMDP`. Note that the policies are not necessarily the same.

d. [2 points] Sometimes, we might reasonably wonder how an optimal policy learned for one MDP might perform if applied to another MDP with similar structure but slightly different characteristics. For example, imagine that you created an MDP to choose an optimal strategy for playing "traditional" blackjack, with a standard card deck and a threshold of 21. You're living it up in Vegas every weekend, but the casinos get wise to your approach and decide to make a change to the game to disrupt your strategy: going forward, the threshold for the blackjack tables is 17 instead of 21. If you continued playing the modified game with your original policy, how well would you do? (This is just a hypothetical example; we won't look specifically at the blackjack game in this problem.)

We have provided a function in the grader `4d-helper` which does the following: (1) First, the function runs value iteration on the `originalMDP` to compute an optimal policy for that MDP. Then, the optimal policy for `originalMDP` is fixed and simulated on `newThresholdMDP`. (2) Then, the function simulates Q-learning directly on `newThresholdMDP` with `blackjackFeatureExtractor` and the default exploration probability. What is the expected (average) rewards from the simulation in (1)? What is the expected (average) rewards under the new Q-learning policy in (2)? Provide some explanation for how the rewards compare, and why they are different.

What we expect: One sentence giving the approximated expected rewards for the transferred `originalMDP` and the directly learned Q-learning policy on the `modifiedMDP` and which policy is better. 2-3 sentences explaining why you think the one policy is better than the other.

Your Solution: The average total rewards for applying the policy found by Value Iteration from originalMDP to modifiedMDP is approximately 6.8. The average total rewards for Qlearning on the modifiedMDP approximately 9.5 (the average total rewards changes slightly with each simulation run).

The QLearning policy is better on modifiedMDP because:

- In general, optimal policies in one environment aren't necessarily optimal in another environment.
- Value Iteration finds the optimal policy for a 'fixed' environment (as modelled by an MDP). So when the environment changes the solution isn't generalisable.

- QLearning learns the optimal actions without prior knowledge about the environment (no MDP model required / model free) so we should cautiously expect it to perform better than a static policy carried from originalMDP to modifiedMDP when the threshold gets changed from 21 to 17.

Problem 5: Modeling Sea Level Rise

Sometimes the skills we learn by playing games can serve us well in real life. In this assignment, you've created a MDP that can learn an effective policy for Blackjack. Now let's see how an MDP can help us make decisions in a scenario with much higher stakes: climate change mitigation strategy.[1] Climate change can cause sea level rise, higher tides, more frequent storms, and other events that can damage a coastal city. Cities want to build infrastructure to protect their citizens, such as seawalls or landscaping that can capture storm surge, but have limited budgets. For this problem, we have implemented an MDP `SeaLevelRiseMDP` in `submission.py` that models how a coastal city government adapts to rising sea levels over the course of multiple decades. There are 2 actions available to the government at each timestep:

- a_{Invest} - Invest in infrastructure during this budget cycle
- a_{Wait} - Hold off in investing in infrastructure and save your surplus budget

Every simulation starts out in the year **2000** and with an initial sea level of **0** (in centimeters). The initial amount of money (in millions of USD), initial amount of infrastructure (unitless), and number of years to run the simulation for are parameters to the model. Every **10 years**, the city government gets a chance to make an infrastructure decision. If the city government *chooses to invest* in infrastructure for that 10 year cycle, then **the current infrastructure state is incremented by 3** and **current budget decreases by $2 mil**. If the city government government *chooses to wait* and not invest this cycle, then the infrastructure state remains the same and **the budget increases by $2 mil**. Typically, **no reward is given until the end of the simulation**. However, if discounting is being applied, then at each time step the **current budget** is given as reward.

However, the budget is not the only thing increasing in our simulation. At each 10 year timestep, the sea level rises by a non-deterministic amount. Specifically, it can **rise a little (1 cm.), a moderate amount (2 cm.), or a lot (3 cm.)** similar to the IPCC sea level rise projection.[2] A moderate rise in sea level is the most likely at each timestep (50% probability), but there is some probability a less or more extreme rise could occur (25% each). Normally, so long as the current sea level below the current infrastructure level, the city can go about business as usual with no punishment. However, if at any point the current sea level surpasses the current infrastructure, then **the city is immediately flooded**, the simulation ends, and the city incurs a large negative reward, simulating the cost of the city becoming uninhabitable.

The threat of sea level is not equal across coastal cities, however. For example, Tampa

[1] https://proceedings.mlr.press/v119/shuvo20a.htmlShuvo et al. A Markov Decision Process Model for Socio-Economic Systems Impacted by Climate Change. ICML. 2020.

[2] IPCC AR6 Sea Level Projection Tool. IPCC 6th Assessment Report. 2021.

Bay, FL also experiences hurricanes regularly, and rising sea levels significantly exacerbate the damage caused by these extreme weather events.[3] In order to better model cities vulnerable to extreme weather events, we can toggle a boolean `disaster`. When `True`, at each time step there is a small possibility that the city is immediately flooded by a natural disaster, which is higher when the sea level is close to the infrastructure level and lower when the sea level is much lower than the infrastructure level. If the city manages to avoid being flooded by the sea until the final year of simulation, then the **current budget is given as reward** and the simulation is ended. However, if the sea level has overtaken the city's infrastructure in this final year, the city does **not** receive the reward and receives the same negative reward as before. Using this MDP, you will explore explore how the optimal policy changes under different scenarios and reason about the strengths and drawbacks of modeling social decisions with machine learning and search.

[3]https://www.nature.com/articles/s41467-019-11755-z Marsooli et al. Climate change exacerbates hurricane flood hazards along US Atlantic and Gulf Coasts in spatially varying patterns. Nature Communications. 2019.

a. [2 points] Imagine you're on L.A.'s city council and you're interested in understanding how sea level rise will impact the city in the coming years. To do so, you will compare 2 simple setups of the `SeaLevelRiseMDP`: one where the simulation is run for **40 years** and one where the simulation is run for **100 years**. We have already implemented functions with the proper parameters to do this with you; all you need to do is run grader test **5a-helper** to examine the optimal policy for these two MDPs. Note that the only parameter difference between the two MDPs is **the number of years to run the simulation for**. Looking at only a 40 year time horizon, interpret the MDP's optimal sequence of actions into the most economical plan for the city government to take. What about for a 100 year time horizon? Using your understanding of the MDPs, why do you think the two MDPs recommended these different economic policies?

What we expect: A 1-2 sentence indication of what series of action/s are economically preferable for both the 40 year horizon MDP and the 100 year horizon MDP. A 1-2 sentence explanation as to why this would be the case. Note: you do not need to write out a full action sequence for the entire simulation - just a general description of what the most economic policy is will suffice.

Your Solution: It is economically preferable to not invest at all over the 40 year horizon MDP. It is economically preferable to invest during the 100 year horizon MDP.

The 40 year MDP outputs a ratio of 0 invest to wait states, which means there's no investment actions. The 100 year MDP outputs a ratio of 0.4 invest to wait states. Reformulating this to be the ratio of invest actions vs. all ations, we arrive: 3292 invest states /(3292 invest states + 7708 wait states) = 0.3 invest actions vs. all actions, which implies investing on average once every 33 years.

In the 40 year horizon, the wait policy is optimal because the initial infrastructure level is 13 and the sea level will never exceed this even in the worst case of 3cm/decade x 4 decades = 12 cm. On the other-hand, in the 100 year horizon, the wait policy is not optimal because the sea level is likely to exceed the infrastructure level so investing is required to maximise rewards.

b. [2 points] In addition to the economic reasons for choosing one economic plan over another that you just explored, there are ethical considerations as well. Consider the following arguments:

- **Veil of Ignorance**: Imagine that you did not know what year you would be born and therefore what state of climate change you would be experiencing. The policy you would choose on that basis would be the fairest policy, as it is the one you would subject yourself to if you did not know what decade you would be born.[4]
- **Uncertain Future**: The future is uncertain. We should not sacrifice present well-being or consumption to ward off a future that might or might not happen.
- **Precautionary Risks**: People have different attitudes towards risk and uncertainty. Doing nothing now represents a significant risk of degraded living conditions for those who will be alive in the future. Since we do not know the risk-tolerance of future people, we ought to act conservatively on their behalf.[5]
- **Symmetry of Future Generations**: People born today and people born in 50 years are equally morally important and deserving of well-being. Privileging the well-being of our generation would be arbitrary; instead we should give their interests and ours equal consideration.
- **Vulnerability of Future Generations**: The well-being of people in future generations is completely dependent on the environmental choices we make today, yet they have no say in those choices. There is no morally relevant reason why we get to make those choices for them; we just happened to be born earlier. Thus we should give their interests and ours equal consideration.[6]

Using your answer for 5a, what investment policy would you advocate for the L.A. city government to use to decide its economic agenda for the next 50 years? Support your argument using one of the above ethical considerations, or another one with proper definition and citation.

What we expect: 2-3 sentences advocating for predicted economic policy, justified with one of the ethical considerations above (or self-defined, with references). Be sure to explicitly state which MDP you chose and reiterate the optimal investment strategy this MDP suggests. Also be sure to explicitly state which ethical argument you are using in your answer for full credit.

Your Solution: For L.A. city government's next 50 year economic agenda I advocate an investment economic policy based on the **Vulnerability of Future Generations** argument. It's irresponsible and reckless to not act if we know the long term effects

[4]https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-9833.2009.01447.x Moellendorf et al. Justice and the Assignment of the Intergenerational Costs of Climate Change.

[5]https://academic.oup.com/monist/article-abstract/102/1/66/5255709 Buchak et al. Weighing the Risks of Climate Change

[6]https://proceedings.mlr.press/v119/shuvo20a.html Shuvo et al. A Markov Decision Process Model for Socio-Economic Systems Impacted by Climate Change. ICML. 2020.

of humanity's impact on the world. However, I don't equally weight the experiences of unborn future generations with those who are presently living. Present suffering by conscious people is weighted higher than future generations who aren't yet experiencing any suffering.

As a result, I base my advocacy for an investment economic policy on the 100-year time horizon MDP which recommends investing.

c. [2 points] Not all cities have as consistent weather as L.A. Imagine instead that you're on the city council for Tampa, Florida - a city almost as well known for its hurricanes as it is its beaches. The Tampa city council is making a centennial plan for the city. Recent climate models indicate that the sea level may rise as much as 200 cm over the next century, with much uncertainty. A friend of yours on L.A city council has recommended using the same MDP from above to help plan your actions.
To adapt the MDP for Tampa, we need to model an increase in devastating weather events like hurricanes in addition to the steady rise of sea level. You will compare 2 similar setups of the `SeaLevelRiseMDP` where there is a small chance of major disaster every 10 years. The key difference between these two MDPs is that one **discounts future rewards**,[7] while the other considers rewards from all states equally. We have already implemented functions with the proper parameters to do this with you; all you need to do is run grader test `5c-helper` to examine the optimal policy for these two MDPs. Note that the only parameter difference between the two MDPs is the discount factor. How much infrastructure should you be considering adding in your centennial plan according to the discounted model? What about for the non-discounted model? Using your understanding of the MDP, why do you think the two MDPs recommended these actions?

What we expect: A 1-2 sentence indication of how much infrastructure additions both the discounted and non-discounted MDPs recommend. A 1-2 sentence explanation as to why the different strategies would be recommended by the two MDPs. Note: you do not need to quantify anything, just simply a general explanation of a policy will do.

Your Solution: For discount factor 0.7 over 100 years the Value Iteration MDP for Tampa recommends a ratio of 0.4 invest to wait states. On the other hand, for discount factor 1 the Value Iteration MDP for Tampa recommends a ratio of 1.3 invest to wait states. This translates to investing on average:

- MDP (Discount factor 0.7): 3141 invest states /(3141 invest states+ 7228 wait states) = 0.30 (once every 33 years).
- MDP (Discount factor 1.0): 6032 invest states /(6032 invest states + 4626 wait states) = 0.56 (once every 18 years).

The MDPs recommend different strategies because of the effect of the discount factor. Value Iteration employs the Bellman equation which is an iterative algorithm for $t = 1, ..., t_{VI}$. At each iteration the discount factor is applied the previous $V_{Opt}^{t-1}(s')$. Value will bias the present rather than future so it means deferring action to later, hence MDP with discount factor 0.7 recommends investing once every 33 years on average vs. once every 18 years for no discount.

[7] https://grist.org/article/discount-rates-a-boring-thing-you-should-know-about-with-otters/ Discount Rates: a boring thing you should know about.

d. [2 points] Finally, remember in question 4d we compared how well (or poorly!) one MDP's optimal policy would transfer to another. If we're no longer quite as good at playing blackjack, that's one thing. However, in a high-stakes scenario like this, supposedly optimal policies that are in fact suboptimal can have severe consequences that affect the livelihoods and well-being of potentially millions of people.
We will now explore what happens when we mis-estimate the cost of climate change. Imagine you're still on Tampa's city council. Recently, the city was hit by a small hurricane which flooded a few parts of the outskirts of the city. The estimated cost of the flooding was -$10 million. You decide to run your sea level rise MDP with this as the negative reward and you present the optimal policy to city council, which decides to use the MDP's policy to set their infrastructure economic agenda for the foreseeable future.

Part 1.) Run `5d-helper` to output the expected reward of the optimal policy from ValueIteration running the `SeaLevelRiseMDP` for 100 years with a flooding cost of -$10 million.

Part 2.) Now, let's imagine that this flooding cost underestimates the cost of flooding by 2 orders of magnitude (read: a flooding cost of -$10 billion). Next in the output from `5d-helper`, you'll find the expected reward of the above fixed optimal policy re-run on an MDP with the more realistic flooding cost.

Part 3.) Finally, you will see the list of actions predicted by the optimal policy for the -$10 mil. flooding cost MDP and the optimal policy for the -$10,000 mil flooding cost MDP.

Given these findings, what do you advise city council to do in regards to their infrastructure economic agenda?

What we expect: 1-2 sentences discussing whether or not you think the city council should still use the -$10 mil. flooding cost MDP to make infrastructure economic decisions. If you think city council should still use the model, provide justification through either the data from `5d-helper` or other outside sources. If you think city council should not use the model, suggest an alternate way city council can still safely incorporate the predictions from `SeaLevelRiseMDP` into their economic decisions.

Your Solution: The Tampa council should not use the $10 million flooding cost MDP to base their infrastructure economic decisions over the next 100 years. This cost figure fails to incorporate the increasing intensity and frequency of flooding effects due to climate change over time and the increasing value of infrastructure - we should expect these to all increase over time.

An alternate way city council can still safely incorporate the predictions from SeaLevelRiseMDP with $10 million flooding cost in their economic decisions is to use it to

provoke further discussions. This MDP can be safely used to pose the question to council members: *what must be true for the MDP with $10 million flooding cost to be valid for the next 100 years?*. For this to be true:

- population doesn't increase
- the number of infrastructure doesn't increase
- the value of that infrastructure doesn't increase

All the above is improbable. And that's without consideration for climate change. This should provoke action to improve the MDP model to guide infrastructure economic decisions.