XCS221 Assignment 3 — Peeking Blackjack

Due Sunday, Dec. 11 at 11:59pm PT.

Guidelines

1. If you have a question about this homework, we encourage you to post your question on our Slack channel, at http://xcs221-scpd.slack.com/
2. Familiarize yourself with the collaboration and honor code policy before starting work.
3. For the coding problems, you must use the packages specified in the provided environment description. Since the autograder uses this environment, we will not be able to grade any submissions which import unexpected libraries.

Submission Instructions

Written Submission: Some questions in this assignment require a written response. For these questions, you should submit a PDF with your solutions online in the online student portal. As long as the PDF is legible and organized, the course staff has no preference between a handwritten and a typeset LaTeX submission. If you wish to typeset your submission and are new to LaTeX, you can get started with the following:

- Type responses only in `submission.tex`.
- Submit the compiled PDF, **not** `submission.tex`.
- Use the commented instructions within the `Makefile` and `README.md` to get started.

Also note that your answers should be in order and clearly and correctly labeled to receive credit. Be sure to submit your final answers as a PDF and **tag all pages correctly when submitting to Gradescope**.

Coding Submission: Some questions in this assignment require a coding response. For these questions, you should submit only the `src/submission.py` file in the online student portal. For further details, see Writing Code and Running the Autograder below.

Honor code

We strongly encourage students to form study groups. Students may discuss and work on homework problems in groups. However, each student must write down the solutions independently, and without referring to written notes from the joint session. In other words, each student must understand the solution well enough in order to reconstruct it by him/herself. In addition, each student should write on the problem set the set of people with whom s/he collaborated. Further, because we occasionally reuse problem set questions from previous years, we expect students not to copy, refer to, or look at the solutions in preparing their answers. It is an honor code violation to intentionally refer to a previous year's solutions. More information regarding the Stanford honor code can be foudn at https://communitystandards.stanford.edu/policies-and-guidance/honor-code.

Writing Code and Running the Autograder

All your code should be entered into `src/submission.py`. When editing `src/submission.py`, please only make changes between the lines containing `### START_CODE_HERE ###` and `### END_CODE_HERE ###`. Do not make changes to files other than `src/submission.py`.

The unit tests in `src/grader.py` (the autograder) will be used to verify a correct submission. Run the autograder locally using the following terminal command within the `src/` subdirectory:

```
$ python grader.py
```

There are two types of unit tests used by the autograder:

- `basic`: These tests are provided to make sure that your inputs and outputs are on the right track, and that the hidden evaluation tests will be able to execute.
- `hidden`: These unit tests are the evaluated elements of the assignment, and run your code with more complex inputs and corner cases. Just because your code passed the basic local tests does not necessarily mean that they will pass all of the hidden tests. These evaluative hidden tests will be run when you submit your code to the Gradescope autograder via the online student portal, and will provide feedback on how many points you have earned.

For debugging purposes, you can run a single unit test locally. For example, you can run the test case `3a-0-basic` using the following terminal command within the `src/` subdirectory:

```
$ python grader.py 3a-0-basic
```

Before beginning this course, please walk through the Anaconda Setup for XCS Courses to familiarize yourself with the coding environment. Use the env defined in `src/environment.yml` to run your code. This is the same environment used by the online autograder.

Test Cases

The autograder is a thin wrapper over the python `unittest` framework. It can be run either locally (on your computer) or remotely (on SCPD servers). The following description demonstrates what test results will look like for both local and remote execution. For the sake of example, we will consider two generic tests: `1a-0-basic` and `1a-1-hidden`.

Local Execution - Hidden Tests

All hidden tests rely on files that are not provided to students. Therefore, the tests can only be run remotely. When a hidden test like `1a-1-hidden` is executed locally, it will produce the following result:

```
----- START 1a-1-hidden:  Test multiple instances of the same word in a sentence.
----- END 1a-1-hidden [took 0:00:00.011989 (max allowed 1 seconds), ???/3 points] (hidden test ungraded)
```

Local Execution - Basic Tests

When a basic test like `1a-0-basic` passes locally, the autograder will indicate success:

```
----- START 1a-0-basic:  Basic test case.
----- END 1a-0-basic [took 0:00:00.000062 (max allowed 1 seconds), 2/2 points]
```

When a basic test like `1a-0-basic` fails locally, the error is printed to the terminal, along with a stack trace indicating where the error occurred:

```
----- START 1a-0-basic:  Basic test case.
<class 'AssertionError'>
{'a': 2, 'b': 1} != None
  File "/Users/grinch/Local_Documents/Software/anaconda3/envs/XCS221/lib/python3.6/unittest/case.py", line 59, in testPartExecutor
    yield
  File "/Users/grinch/Local_Documents/Software/anaconda3/envs/XCS221/lib/python3.6/unittest/case.py", line 605, in run
    testMethod()
  File "/Users/grinch/Local_Documents/SCPD/XCS221/A1/src/graderUtil.py", line 54, in wrapper
    result = func(*args, **kwargs)
  File "/Users/grinch/Local_Documents/SCPD/XCS221/A1/src/graderUtil.py", line 83, in wrapper
    result = func(*args, **kwargs)
  File "/Users/grinch/Local_Documents/SCPD/XCS221/A1/src/grader.py", line 23, in test_0
    submission.extractWordFeatures("a b a"))
  File "/Users/grinch/Local_Documents/Software/anaconda3/envs/XCS221/lib/python3.6/unittest/case.py", line 829, in assertEqual
    assertion_func(first, second, msg=msg)
  File "/Users/grinch/Local_Documents/Software/anaconda3/envs/XCS221/lib/python3.6/unittest/case.py", line 822, in _baseAssertEqual
    raise self.failureException(msg)
----- END 1a-0-basic [took 0:00:00.003809 (max allowed 1 seconds), 0/2 points]
```


Remote Execution

Basic and hidden tests are treated the same by the remote autograder. Here are screenshots of failed basic and hidden tests. Notice that the same information (error and stack trace) is provided as the in local autograder, now for both basic and hidden tests.

1a-0-basic) Basic test case. (0.0/2.0)

```
<class 'AssertionError'>:  {'a': 2, 'b': 1} != None
  File "/autograder/source/miniconda/envs/XCS221/lib/python3.6/unittest/case.py", line 59, in testPartExecutor
    yield
  File "/autograder/source/miniconda/envs/XCS221/lib/python3.6/unittest/case.py", line 605, in run
    testMethod()
  File "/autograder/source/graderUtil.py", line 54, in wrapper
    result = func(*args, **kwargs)
  File "/autograder/source/graderUtil.py", line 83, in wrapper
    result = func(*args, **kwargs)
  File "/autograder/source/grader.py", line 23, in test_0
    submission.extractWordFeatures("a b a"))
  File "/autograder/source/miniconda/envs/XCS221/lib/python3.6/unittest/case.py", line 829, in assertEqual
    assertion_func(first, second, msg=msg)
  File "/autograder/source/miniconda/envs/XCS221/lib/python3.6/unittest/case.py", line 822, in _baseAssertEqual
    raise self.failureException(msg)
```

Just like in the local autograder, this error caused the test to fail.

Just like in the local autograder, start your debugging in line 23 of grader.py.

1a-1-hidden) Test multiple instances of the same word in a sentence. (0.0/3.0)

```
<class 'AssertionError'>:  {'a': 23, 'ab': 22, 'aa': 24, 'c': 16, 'b': 15} != None
  File "/autograder/source/miniconda/envs/XCS221/lib/python3.6/unittest/case.py", line 59, in testPartExecutor
    yield
  File "/autograder/source/miniconda/envs/XCS221/lib/python3.6/unittest/case.py", line 605, in run
    testMethod()
  File "/autograder/source/graderUtil.py", line 54, in wrapper
    result = func(*args, **kwargs)
  File "/autograder/source/graderUtil.py", line 83, in wrapper
    result = func(*args, **kwargs)
  File "/autograder/source/grader.py", line 31, in test_1
    self.compare_with_solution_or_wait(submission, 'extractWordFeatures', lambda f: f(sentence))
  File "/autograder/source/graderUtil.py", line 183, in compare_with_solution_or_wait
    self.assertEqual(ans1, ans2)
  File "/autograder/source/miniconda/envs/XCS221/lib/python3.6/unittest/case.py", line 829, in assertEqual
    assertion_func(first, second, msg=msg)
  File "/autograder/source/miniconda/envs/XCS221/lib/python3.6/unittest/case.py", line 822, in _baseAssertEqual
    raise self.failureException(msg)
```

This error caused the test to fail.

Start your debugging in line 31 of grader.py.

Finally, here is what it looks like when basic and hidden tests pass in the remote autograder.

1a-0-basic) Basic test case. (2.0/2.0)

1a-1-hidden) Test multiple instances of the same word in a sentence. (3.0/3.0)

Introduction

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.

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 {-2, -1, 0, 1, 2}. You start at state 0, and if you reach either -2 or 2, the game ends. At each state, you can take one of two actions: {-1, +1}.

If you're in state s and choose -1:

- You have an 80% chance of reaching the state $s - 1$.
- You have a 20% chance of reaching the state $s + 1$.

If you're in state s and choose +1:

- You have a 30% chance of reaching the state $s + 1$.
- You have a 70% chance of reaching the state $s - 1$.

If your action results in transitioning to state -2, then you receive a reward of 20. If your action results in transitioning to state 2, then your reward is 100. Otherwise, your reward is -5. Assume the discount factor γ is 1.

(a) **[4 points (Written)]** Give the value of $V_{\text{opt}}(s)$ for each state s after 0, 1, and 2 iterations of value iteration. Iteration 0 just initializes all the values of V to 0. Terminal states do not have any optimal policies and take on a value of 0.

(b) **[4 points (Written)]** Based on $V_{\text{opt}}(s)$ after the second iteration, what is the corresponding optimal policy π_{opt} for all non-terminal states?

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 (Written)]**

Suppose we have an MDP with states S and a discount factor $\lambda < 1$, 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?

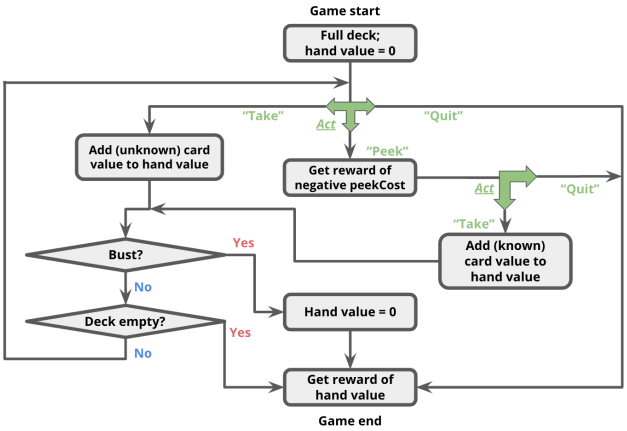
Let us define a new MDP with states $S' = S \cup \{o\}$, where o is a new state. Let's use the same actions ($A'(s) = A(s)$), but we need to keep the discount $\lambda' = 1$. Your job is to define new transition probabilities $T'(s, a, s')$ and rewards $R'(s, a, s')$ in terms of the old MDP such that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

Hint: If you're not sure how to approach this problem, go back to the first MDP lecture and read closely the slides on convergence.

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.

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. Each round, the player either (i) takes the next card from the top of the deck (costing nothing), (ii) peeks at the top card (costing `peekCost`, in which case the next round, that card will be drawn), or (iii) quits the game. (Note: it is not possible to peek twice in a row; if the player peeks twice in a row, then `succAndProbReward()` should return `[]`.)

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

- The player quits, in which case her reward is the sum of the face values of the cards in her hand.
- The player takes a card 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 quits, 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.*

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 1, and the threshold is 4. Initially, the player has no cards, so her total is 0; this corresponds to state `(0, None, (1, 1, 1))`. At this point, she can take, peek, or quit.

- If she takes, the three possible successor states (each of which has equal probability of 1/3) are:

  ```
  (1, None, (0, 1, 1))
  (2, None, (1, 0, 1))
  (3, None, (1, 1, 0))
  ```

  She will receive a reward of 0 for reaching any of these states. (Remember, even though she 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.)

- If she peeks, the three possible successor states are:

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

She will receive (immediate) reward `-peekCost` for reaching any of these states. Things to remember about the states after a peek action:

  - From `(0, 0, (1, 1, 1))`, taking a card will lead to the state `(1, None, (0, 1, 1))` deterministically.
  - 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.

- If she quits, then the resulting state will be `(0, None, None)`. (Remember that setting the deck to `None` signifies the end of the game.)

As another example, let's say the player's current state is `(3, None, (1, 1, 0))`, and the threshold remains 4.

- If she quits, the successor state will be `(3, None, None)`.
- If she takes, the successor states are `(3 + 1, None, (0, 1, 0))` or `(3 + 2, None, None)`.

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

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 and strategy at the same time!

(a) **[8 points (Coding)]** 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 $\hat{Q}_{\text{opt}}(s,a) = \mathbb{W} \cdot \phi(s,a)$, where in code, $\mathbb{W}$ is `self.weights`, ϕ is the `featureExtractor` function, and $\hat{Q}_{\text{opt}}$ 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) **[5 points (Written)]** Now let's apply Q-learning to an MDP and see how well it performs in comparison with value iteration. First, call `simulate` using your Q-learning code and the `identityFeatureExtractor()` on the MDP `smallMDP` (defined for you in `submission.py`), with 30000 trials. 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)? (Don't forget to set the explorationProb of your Q-learning algorithm to 0 after learning the policy.) Now run `simulate()` on `largeMDP`, again with 30000 trials. How does the policy learned in this case compare to the policy learned by value iteration? What went wrong?

(c) **[5 points (Coding)]** 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 on the `largeMDP`.

(d) **[5 points (Written)]** 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.)

To explore this scenario, let's take a brief look at how a policy learned using value iteration responds to a change in the rules of the MDP.

- First, run value iteration on the `originalMDP` (defined for you in `submission.py`) to compute an optimal policy for that MDP.
- Next, simulate your policy on `newThresholdMDP` (also defined for you in `submission.py`) by calling `simulate` with an instance of `FixedRLAlgorithm` that has been instantiated using the policy you computed with value iteration. What is the expected reward from this simulation?
  Hint: read the documentation (comments) for the `simulate` function in util.py, and look specifically at the format of the function's return value.
- Now try simulating Q-learning directly on `newThresholdMDP` with `blackjackFeatureExtractor` and the default exploration probability. What is your expected reward under the new Q-learning policy? Provide some explanation for how the rewards compare, and why they are different.

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 **the 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 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 how the optimal policy changes under different scenarios and reason about the strengths and drawbacks of modeling social decisions with machine learning and search.

(a) **[3 points (Written)]**

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-0` to examine the optimal policy for these two MDPs. Note that the only parameter difference between

[1] Shuvo et al. A Markov Decision Process Model for Socio-Economic Systems Impacted by Climate Change. ICML. 2020.
[2] IPCC. IPCC AR6 Sea Level Projection Tool. IPCC 6th Assessment Report. 2021.
[3] Marsooli et al. Climate change exacerbates hurricane flood hazards along US Atlantic and Gulf Coasts in spatially varying patterns. Nature Communications. 2019.

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.

(b) **[3 points (Written)]**

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.

(c) **[3 points (Written)]**

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

[4] Moellendorf et al. Justice and the Assignment of the Intergenerational Costs of Climate Change.
[5] Buchak et al. Weighing the Risks of Climate Change.
[6] Shuvo et al. A Markov Decision Process Model for Socio-Economic Systems Impacted by Climate Change. ICML. 2020.

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-0` 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.

(d) **[3 points (Written)]**

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-0` 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 3 orders of magnitude (read: a flooding cost of -$10 billion). Next in the output from `5d-helper-0`, 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-0` 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.

[7] Discount Rates: a boring thing you should know about.

This handout includes space for every question that requires a written response. Please feel free to use it to handwrite your solutions (legibly, please). If you choose to typeset your solutions, the `README.md` for this assignment includes instructions to regenerate this handout with your typeset LaTeX solutions.

1.a

1.b

2.a

4.b

4.d

5.a

5.b

5.c

5.d