CS 5180 Exercise 3

Harin Kumar Nallaguntla

Study Partners: Gopi Sainath Mamindlapalli

1. a.

$$v_*(s) = \max q_*(s,a)$$

b.

$$q_*(s,a) = \max p(s', r|s,a)v_*(s)$$

c.

$$\pi(a|s) = \max_a q_*(s,a)$$

d.

$$\pi(a|s) = \max_a \sum_{s',r} p(s', r|s,a)[r + \gamma v_*(s)]$$

e.

$$\begin{aligned}
&\mathrm{v}_\pi(s) = \textstyle\sum_a \pi(a|s) \sum_{s'} p(s'|s,a)[r(s,a) + \gamma v_\pi(s')] \\
&\mathrm{v}_*(s) = \max_a \textstyle\sum_{s'} p(s'|s,a)[r(s,a) + \gamma v_*(s')] \\
&\mathrm{q}_\pi(s,a) = \textstyle\sum_{s'} p(s'|s,a)[r(s,a) + \gamma \sum_{a'} \pi(a'|s')q_\pi(s',a')] \\
&\mathrm{q}_*(s,a) = \textstyle\sum_{s'} p(s'|s,a)[r(s,a) + \gamma \max_{a'} q_*(s',a')]
\end{aligned}$$

2.a. The policy iteration algorithm presents a subtle issue that arises when considering a situation in which we find ourselves in state s, and both actions a1 and a2 specified by the policy π(a|s) lead to the same state s` (assuming s` is a terminal state and there are multiple paths to reach it). In such a scenario, the policy can continually alternate between these actions, potentially preventing the algorithm from reaching convergence. The problematic aspect of the algorithm lies in the second-to-last line, where it checks if the old action is not equal to the new action π(s) and subsequently sets the "policy-stable" variable to "fails." This check fails to address the issue of oscillation between equally advantageous policies.

To address this challenge, it is essential to replace this line with a more robust condition. Instead of merely verifying the inequality between old and new actions, we should compare the values they produce. If these values are the same, it signifies policy stability. The modified condition reads as follows: **If $v_{pi`}(s) = v_{pi}(s)$, then "policy-stable" should be set to "true."** This change ensures that the algorithm can handle the issue of policy oscillation and converge more effectively.

b. Value iteration offers a solution to the problem of policy oscillation found in policy iteration, as it doesn't explicitly alternate between actions. However, value iteration can still experience value oscillations when actions have similar value estimates. To address this, a common strategy is to introduce a small positive constant ϵ and use a stopping condition that checks if the difference between consecutive value estimates is less than ϵ for all states.

This ensures that the value estimates have stabilized within a small range and helps prevent unnecessary oscillation in the value function. The choice of ϵ depends on the specific problem and the desired level of precision. In summary, while value iteration may encounter value oscillations, it differs from policy oscillation in policy iteration, and employing the mentioned approach can mitigate these oscillations and eventually lead to convergence to the optimal value function. Mathematically,

If $|V_{k+1}(s) - V_k(s)| < \epsilon$ for all states s, then halt and provide V_{k+1}

3.a. Initialization:
Q(s, a) ϵ R^2 and and π(s) ϵ A(s) arbitrarily for all s ϵ S;

Policy Evaluation:

```
$\Delta \leftarrow 0$
$foreachs \in S:$
$foreacha \in A_s:$
$q(s,a) \leftarrow Q(s,a)$
$Q(s,a) \leftarrow \sum_{s',r} p(s',r|s,a)[r + \gamma \sum_{a'} \pi(a'|s')Q(s',a')]$
$\Delta \leftarrow \max(\Delta, |q(s,a) - Q(s,a)|)$
$repeatuntil\Delta < \theta(asmallpositivethreshold)$
```

Policy Improvement:

```
$policy - stable \leftarrow true$
$foreachs \in S:$
$old - action \leftarrow \pi(s)$
$\pi(s) \leftarrow \arg\max_a Q(s,a)$
$ifold - action \neq \pi(s):$
$policy - stable \leftarrow false$
$ifpolicy - stable = returnQ^*$
```

b.

$$q_{k+1}(s,a) = E\left[R_{t+1} + \max_{a_0} \gamma q_k(s',a')\right] = \sum_{s',r} p(s_0,r|s,a)\left(r + \max_{a'} \gamma q_k(s',a')\right)$$

4.a. The agent wants to reach a specific place called "state z" as quickly as possible because it costs them something for every moment they spend in two other places, "state x" and "state y." However, there's a problem. The action that can take them to "state z" (let's call it "action c") doesn't work very well - it doesn't succeed often.
So, the agent's job is to figure out how to get to "state z" while spending as little as possible on the way. Here's what it should do:

1. In "state x," it should try "action c" because that's the best way to reach "state z."
2. In "state y," it might be smarter to start with "action b" to go back to "state x" (where it's better to wait for a chance to get to "state z") instead of trying to go directly to "state z." This decision in "state y" is a bit tricky because it involves balancing the cost of waiting with the higher chance of eventually getting to "state z"

b.

Iteration - 1:

Value Iteration:

$$v_x = -1 + 0.15v_z + 0.85v_x = -6.66$$

$$v_y = -2 + 0.15v_z + 0.85v_y = -13.32$$

$$v_z = 0$$

Policy Update:

for state x:

$$\pi(b|x) = 0.8v_y + 0.2v_x = -11.99$$

$$\pi(c|x) = 0.15v_z + 0.85v_x = -5.66$$

for state y:

$$\pi(b|y) = 0.8v_x + 0.2v_y = -7.99$$

$$\pi(c|y) = 0.15v_z + 0.85v_y = -11.32$$

so, for state x, action c is preferred and for state y, action b is preferred
here, *unchanged* is false, so we proceed

Iteration - 2:

Value Iteration:

$$v_x = -1 + 0.15v_z + 0.85v_x = -6.66$$

$$v_y = -2 + 0.8v_x + 0.2v_y = -9.16$$

$$v_z = 0$$

Policy Update:

for state x:

$$\pi(b|x) = 0.8v_y + 0.2v_x = -8.66$$

$$\pi(c|x) = 0.15v_z + 0.85v_x = -5.66$$

for state y:

$$\pi(b|y) = 0.8v_x + 0.2v_y = -7.16$$

$$\pi(c|y) = 0.15v_z + 0.85v_y = -7.79$$

optimal policy: so, for state x, action c is preferred and for state y, action b is preferred

unchanged is **true** in case, so we terminate the iterations and what we obtainained at the end is the optimal policy

c. Starting with an initial plan where action "a" is taken in both states, we encounter a problem that cannot be solved. The problem can be described by these equations:

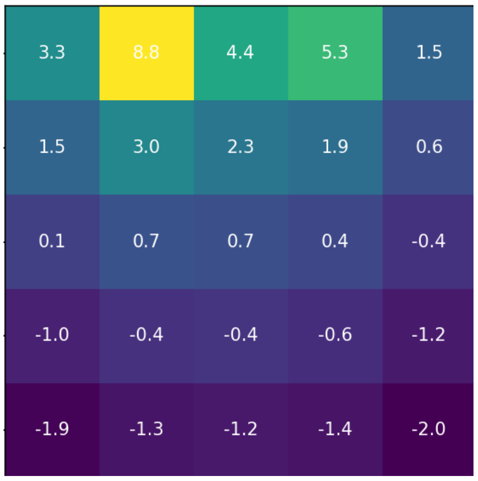
1. The first equation: "vx = -1 + 0.2vx + 0.8vy"
2. The second equation: "vy = -2 + 0.8vx+ 0.2vy"
3. The third equation: "vz = 0"

The first two equations do not match up, meaning they are inconsistent. We cannot solve for the values of vx and vy.

To resolve this issue, we introduce a concept called "discounting." It helps us find clear solutions by limiting the penalty (the expected cost that an agent can face) in both states. However, the choice of the discount factor affects the policy we end up with.

When the discount factor (denoted as γ) is small, it means that the cost in the distant future doesn't have a significant impact on our calculations because γ raised to a high power becomes nearly zero. Consequently, in such cases, an agent might decide to take action "b" in state y. This is because the short-term cost, when discounted, outweighs the long-term cost associated with repeatedly taking action "b" and remaining in state

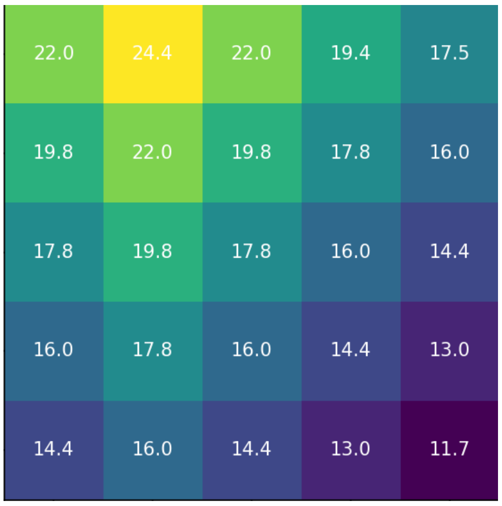
5. a. value function for equiprobable random policy:



3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.6
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

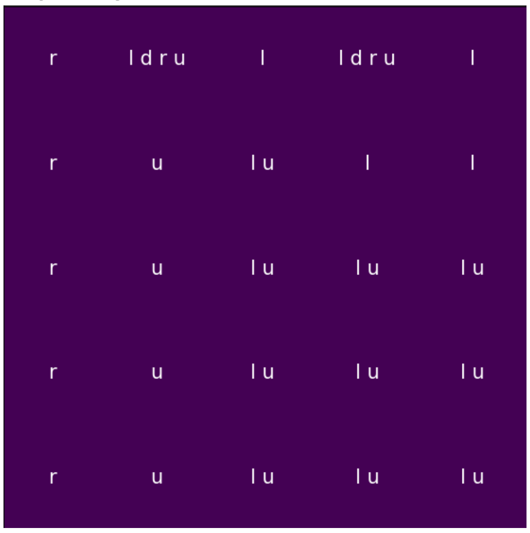
b. the value iteration algorithm ran for 20 iterations

optimal value function:



22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

optimal policy:



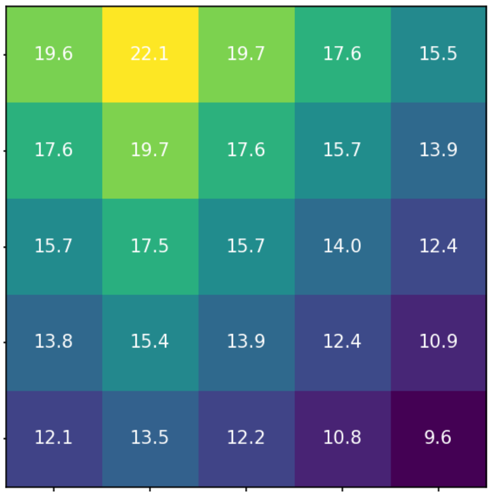
r	l d r u	l	l d r u	l
r	u	l u	l	l
r	u	l u	l u	l u
r	u	l u	l u	l u
r	u	l u	l u	l u

here, l is left, r is right, u is up and d is down

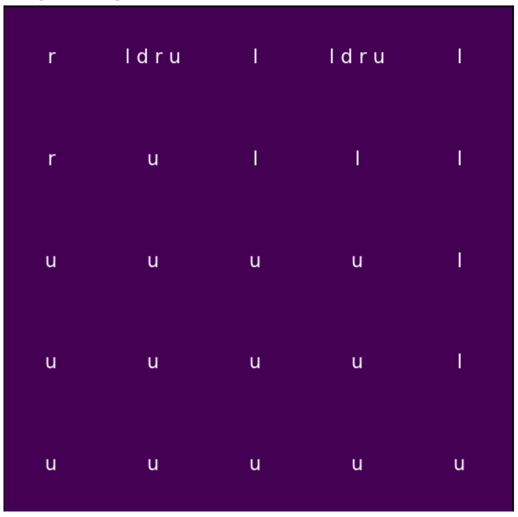
I looked into why the left-most column does not have the up action along with right action and as it turns out, the calculated return for the right action is larger than that of the up action by a very small margin so the up action did not end up in the final value function

c. the policy iteration algorithm ran for 7 iterations, which explains the difference in value function and subsequently the difference in policy

optimal value function:



optimal policy:



here, l is left, r is right, u is up and d is down

6. I could not complete the 6th question. But, I tried my best. Please check out the code I wrote for the dynamics class in env.py :)

7.a. Let's prove the given inequality step by step:

We are given two functions, f(a) and g(a). We want to show that:

max_a [f(a) - max_a g(a)] ≤ max_a |f(a) - g(a)|

To prove this inequality, we'll consider two cases:

1. Case 1: max_a [f(a) - max_a g(a)] ≥ 0
2. Case 2: max_a [f(a) - max_a g(a)] < 0

For both cases, we'll show that the inequality holds.

Case 1: $\max_a [f(a) - \max_a g(a)] \geq 0$

In this case, the left-hand side of the inequality is non-negative.

Now, let's consider the right-hand side:

$\max_a |f(a) - g(a)|$

The absolute value function $|x|$ is always non-negative, so the right-hand side is also non-negative.

Since both sides of the inequality are non-negative, the inequality holds trivially for Case 1.

Case 2: $\max_a [f(a) - \max_a g(a)] < 0$

In this case, the left-hand side of the inequality is negative.

Now, let's consider the right-hand side:

$\max_a |f(a) - g(a)|$

The absolute value function $|x|$ is non-negative, which means that the right-hand side is greater than or equal to 0. In fact, it can only be zero if $f(a)$ and $g(a)$ are identical for all values of 'a.'

Since the left-hand side is negative and the right-hand side is non-negative (and can only be zero if $f(a)$ and $g(a)$ are identical), the inequality also holds for Case 2.

Therefore, for both cases, the inequality holds, and we have shown that:

$\max_a [f(a) - \max_a g(a)] \leq \max_a |f(a) - g(a)|$

This completes the proof.

b.

To prove the given inequality:

$$\|BVi - BV'i\|_\infty \leq \gamma \|Vi - V'i\|_\infty$$

We'll first use the definition of the infinity norm ($\|\cdot\|_\infty$) and then apply the properties of the Bellman backup operator. The infinity norm is defined as:

$$\|x\|_\infty = \max_i |x_i|, whereirangesoverallelementsofthevectorx.$$

Let's break it down step by step:
1. Consider the ith element of $BVi - BV'i$:

$$(BVi - BV'i)_i = \max_a \sum_{s'} p(s'|s,a)[r + \gamma Vi(s')] - \max_a \sum_{s'} p(s'|s,a)[r + \gamma V'i(s')]$$

2. Let's rewrite the above expression:

$$(BVi - BV'i)_i = \max_a \left[\sum_{s'} p(s'|s,a)\gamma(Vi(s') - V'i(s')) \right]$$

3. Now, we apply the infinity norm to both sides:

$$\|(BVi - BV'i)_i\|_\infty = \max_a \left| \sum_{s'} p(s'|s,a)\gamma(Vi(s') - V'i(s')) \right|$$

4. Since this holds for all i, we can take the maximum over i:

$$\|BVi - BV'i\|_\infty = \max_i \max_a \left| \sum_{s'} p(s'|s,a)\gamma(Vi(s') - V'i(s')) \right|$$

5. We can further rewrite the above expression:

$$\|BVi - BV'i\|_\infty = \max_a \|\gamma(Vi - V'i)\|_\infty$$

6. Now, using the definition of the infinity norm, we can express the above as:

$$\|BVi - BV'i\|_\infty = \gamma \max_a \max_i |Vi(s') - V'i(s')|$$

7. Finally, notice that $\max_i |Vi(s') - V'i(s')|$ is the infinity norm of $Vi - V'i$:

$$\|BVi - BV'i\|_\infty = \gamma \|Vi - V'i\|_\infty$$

So, we have shown that:

$$\|BVi - BV'i\|_\infty = \gamma \|Vi - V'i\|_\infty$$

This completes the proof. The inequality holds, and it shows that the infinity norm of the Bellman backup of two value function vectors is bounded by γ times the infinity norm of the difference between the two value function vectors.

c.
To prove that value iteration always converges to the optimal value function v∗ under the assumption that the Bellman backup operator B has a fixed point x (i.e., Bx = x), and assuming $\gamma < 1$, we will show three key points:

1. Value Iteration Converges to the Fixed Point x.
2. The Fixed Point x is Unique.
3. The Unique Fixed Point x is the Optimal Value Function v*.

Let's go through each of these points:

1. Value Iteration Converges to the Fixed Point x:

Since Bx = x by assumption, value iteration can be seen as repeatedly applying the Bellman backup operator B to some initial value function V0. As the number of iterations increases, the sequence of value functions $\{V0, BV0, B^2V0, ...\}$ approaches the fixed point x. This is a fundamental property of value iteration, and it converges to the fixed point.

2. The Fixed Point x is Unique:

To prove uniqueness, assume there are two fixed points x and y, such that Bx = x and By = y. Now, let's consider their difference: z = x - y.

Using the properties of B, we have:

Bz = B(x - y) = Bx - By = x - y = z

This shows that z is also a fixed point of B. However, the assumption was that x and y are fixed points, and z is not unique. Therefore, x and y cannot be distinct fixed points. Hence, the fixed point of B is unique.

3. The Unique Fixed Point x is the Optimal Value Function v:*

Now, we need to show that the unique fixed point x is indeed the optimal value function v*. Suppose it's not, and there exists another value function v' such that v' is optimal (v* is the true optimal value function), and v' ≠ x.

Since $\gamma < 1$, we can apply the result from part (b) that shows that the Bellman backup operator is a contraction by a factor of γ:

$\|Bv - Bv'\|_\infty \leq \gamma \|v - v'\|_\infty$

Now, let's apply this result with v = x (the fixed point) and v' = v* (the true optimal value function):

$\|Bx - Bv\|_\infty \leq \gamma \|x - v\|_\infty$

However, x is a fixed point (Bx = x), so the left-hand side simplifies to:

$\|x - Bv\|_\infty \leq \gamma \|x - v\|_\infty$

Since v* is the optimal value function, we know that $\|Bv^* - v^*\|_\infty = 0$. This means that the right-hand side of the inequality is zero:

$\|x - 0\|_\infty \leq \gamma \|x - v^*\|_\infty$

$\|x\|_\infty \leq \gamma \|x - v^*\|_\infty$

Now, we divide by γ (since $\gamma < 1$):

$\|x\|_\infty / \gamma \leq \|x - v^*\|_\infty$

However, $\|x\|_\infty / \gamma$ is a constant, and the right-hand side, $\|x - v\|_\infty$*, is a measure of the error between the fixed point x and the true optimal value function v*. As the number of iterations of value iteration increases, this error must approach zero. But the only way for this to be possible is if $\|x - v\|_\infty$ *= 0, which means that x = v*.

Therefore, we have shown that the unique fixed point x is indeed the optimal value function v*.

In conclusion, value iteration converges to the unique fixed point, and this unique fixed point is the optimal value function v*, under the assumption that the Bellman backup operator B has a fixed point x and with the condition $\gamma < 1$.