- Suppose someone gives you a randomized policy π in a continuing reinforcement learning programming problem and claims it is optimal. How would you construct a deterministic policy π which is optimal?
  Justify your answer.

  Solution: for every state s, pick any action a such that $\pi(a \mid s) > 0$. Define a new policy that always takes that a. That is a deterministic policy. We will next argue this policy is also optimal.

- To understand why, let us make an observation which will be useful in the next question as well.
- Suppose $\alpha_1, \ldots, \alpha_n$ are nonnegative numbers satisfying $\sum_{i=1}^{n} \alpha_i = 1.$
- We will call $\sum_{i=1}^{n} \alpha_i x_i$ a **convex combination of the numbers** $x_1, \ldots, x_n$**.**
- Claim: a convex combination of numbers is always less than the maximum of the same numbers.

  That is, $\sum_{i=1}^{n} \alpha_i x_i \leq \max_i x_i$
- Proof: let $M = \max_i x_i$. Then

  $$\sum_{i=1}^{n} \alpha_i x_i \leq \sum_{i=1}^{n} \alpha_i M = M \sum_{i=1}^{n} \alpha_i = M$$
- A further observation: the inequality in the last bullet is strict whenever there is some i with $\alpha_i > 0$ and $x_i < M.$
- Consequence: if $\sum_{i=1}^{n} \alpha_i x_i = M$ then every i such that $\alpha_i > 0$ satisfies $x_i = M$

- Now, back to the problem at hand.
- To argue that the new deterministic policy is optimal, consider what happens when you alter the policy π as described before but **one state at a time**.
- That is, enumerate all the states $s_1, \ldots, s_n$.

  First, make $\pi(s_1)$ deterministic as described earlier (without changing $\pi(s_2), \ldots, \pi(s_n)$).

  Then, make $\pi(s_2)$ deterministic.

  Then, make $\pi(s_3)$ deterministic, and so on.
- We next argue: each change still results in an optimal policy.
- This will imply the final policy, which completely deterministic, is optimal.

- Given an optimal π and a state s_i, we have that

$$J_\pi(s_i) = \sum_a \pi(a \mid s_i) \left[r(s_i, a) + \gamma \sum_{s'} P(s' \mid s_i, a) J_\pi(s') \right]$$

and

$$J_\pi(s_i) = \max_a \left[r(s_i, a) + \gamma \sum_{s'} P(s' \mid s_i, a) J_\pi(s') \right]$$

First equation holds for any policy while the second uses the optimality of π

- First equation is a convex combination of the numbers

$$r(s_i, a) + \gamma \sum_{s'} P(s' \mid s_i, a) J_\pi(s')$$

while the second equation is their max.

- ....but the LHS of both equations is the same. So we have here a bunch of numbers whose convex combination is equal to their largest.

- We conclude: all terms with $\pi(a \mid s_i) > 0$ have exactly the same $r(s_i, a) + \gamma \sum_{s'} P(s' \mid s_i, a) J_\pi(s')$
- In other words: all actions chosen by policy π with positive probability have **exactly the same future reward.**
- Thus changing $\pi(a \mid s_i)$ to always choose one of them does not change J_π.

  In other words, the policy remains optimal after making the choice at s_i deterministic.
- **Note:** this was a difficult question, and you did not need to have spelled it out at this level of detail to receive full credit.

- **Note:** you did not need to spell everything out at this level of detail to receive full credit on this question.
- For example, you could have simply asserted that if the optimal policy assigns a positive probability to two actions, the reward-to-go from both actions must be identical. This is intuitive enough that you will get full credit for this. However, if you want to be very precise in arguing for this, you need to make the argument about convex combinations in these solutions.

- Is it always true that $J_\pi(s) \leq \max_a Q_\pi(s,a)$?
- Yes. We know that

  $J_\pi(s) = \sum_a \pi(a \mid s) Q_\pi(s,a)$ (see lecture notes)

- This is a convex combination of the numbers $Q_\pi(s,a)$.
- From the answer to the previous question, a convex combination of a collection of numbers is upper bounded by the max of the same numbers.

- Suppose all the rewards in a continuing reinforcement learning problem are in [0,1] and we have $\gamma = 0.9$.

  As in the lecture notes, we use J^* to denote the rewards-to-go under the optimal policy.

  First, give an upper bound on $||J^*||_\infty$.

  Solution: if all the rewards are super bounded by one and the discount factor is 0.9, then an upper bound on the value function is

  $$1 + 0.9 \cdot 1 + 0.9^2 \cdot 1 + \cdots = \frac{1}{1 - 0.9} = 10$$

  A lower bound is $0 + 0.9 \cdot 0 + 0.9^2 \cdot 0 + \cdots = 0$

  Since every entry of J^* is thus in [0,10], we have that $||J^*||_\infty \leq 10$

  Second, how many iterations does it take until we can guarantee that value iteration produces J_t with $||J_t - J^*||_\infty \leq 0.01$ starting from $J_0 = \mathbf{0}$?

  Solution: from the lecture notes, $||J_t - J^*||_\infty \leq \gamma^t ||J_0 - J^*|| \leq \gamma^t 10$

  So we just need to solve $\gamma^t 10 \leq 0.01$ or

  $0.9^t 10 \leq 0.01$

  This gives $t \geq 65.5$. Since t is an integer, this is the same as $t \geq 66$.

- Look at Eq. (*) in the lecture notes on Q-learning. It was derived under the assumption that the policy π is deterministic. What should the corresponding equation be when the policy is randomized?

  Solution:

$$Q_{t+1}(s,a) = r(s,a) + \sum_{s'} P(s' \mid s,a) \sum_{a'} \pi(a' \mid s') Q_t(s',a')$$

- Consider an MDP with two states, A and B. In A, there are two actions you can take. Action 1 keeps you in state A, with a reward of one. Action 2 moves you to B, with a reward of zero. In state B, there is only one action to take, which keeps you in B with a reward of 2.

  You want to use temporal difference learning to evaluate the rewards-to-go of the "choose at random" policy.
  You generate the following two sample paths:

  $$s_1 = A, a_1 = 1, s_2 = A, a_2 = 2, s_3 = B$$

  $$s_1 = A, a_1 = 2, s_2 = B, a_2 = 1, s_3 = B, a_3 = 1, s_4 = B$$

  Use temporal difference learning to come up with estimates of the rewards-to-go from both states starting from [16,16].

- **Solution:**

$$J_1 = \begin{pmatrix} 16 \\ 16 \end{pmatrix}$$

$$J_2 = \begin{pmatrix} 16 + \frac{1}{1}\left(1 + \frac{1}{2}16 - 16\right) \\ 16 \end{pmatrix} = \begin{pmatrix} 9 \\ 16 \end{pmatrix}$$

$$J_3 = \begin{pmatrix} 9 + \frac{1}{2}\left(0 + \frac{1}{2}16 - 9\right) \\ 16 \end{pmatrix} = \begin{pmatrix} 8.5 \\ 16 \end{pmatrix}$$

- $$J_4 = \begin{pmatrix} 8.5 + \frac{1}{3}\left(0 + \frac{1}{2}16 - 8.5\right) \\ 16 \end{pmatrix} = \begin{pmatrix} 8.3333 \\ 16 \end{pmatrix}$$
$$J_5 = \begin{pmatrix} 8.333 \\ 16 + \frac{1}{4}\left(2 + \frac{1}{2}16 - 16\right) \end{pmatrix} = \begin{pmatrix} 8.3333 \\ 14.5 \end{pmatrix}$$
$$J_5 = \begin{pmatrix} 8.333 \\ 14.5 + \frac{1}{5}\left(2 + \frac{1}{2}14.5 - 14.5\right) \end{pmatrix} = \begin{pmatrix} 8.3333 \\ 13.45 \end{pmatrix}$$
- **Note:** since the problem was stated ambiguously, you will get full credit if you computed the temporal difference process for each sample path separately. But please remember that in the real world you would generate multiple sample paths and keep your iterate from one sample path to the next, as in these solutions.

- Consider an MDP with two states, A and B. In A, there are two actions you can take. Action 1 keeps you in state A, with a reward of one. Action 2 moves you to B, with a reward of zero. In state B, there is only one action to take, which keeps you in B with a reward of 2.

  (i) Suppose you start from [16,16,16]. Starting at state A, write out the first three iterations of Q-learning with $\epsilon = 0$ starting from $s_1 = A, a_1 = 1, s_2 = A, a_2 = 2, s_3 = B, a_3 = 1$

  Solution:

$$Q_1 = \begin{pmatrix} 16 \\ 16 \\ 16 \end{pmatrix}$$

$$Q_2 = \begin{pmatrix} 16 + \frac{1}{1}\left(1 + \frac{1}{2}16 - 16\right) \\ 16 \\ 16 \end{pmatrix} = \begin{pmatrix} 9 \\ 16 \\ 16 \end{pmatrix}$$

$$Q_3 = \begin{pmatrix} 9 \\ 16 + \frac{1}{2}\left(0 + \frac{1}{2}16 - 16\right) \\ 16 \end{pmatrix} = \begin{pmatrix} 9 \\ 12 \\ 16 \end{pmatrix}$$

$$Q_4 = \begin{pmatrix} 9 \\ 12 \\ 16 + \frac{1}{3}\left(2 + \frac{1}{2}16 - 16\right) \end{pmatrix} = \begin{pmatrix} 9 \\ 12 \\ 14 \end{pmatrix}$$

- For code, see https://colab.research.google.com/drive/1uJJLGbjktOdsOr3vA8hb6wXcWCwPxswe?usp=sharing