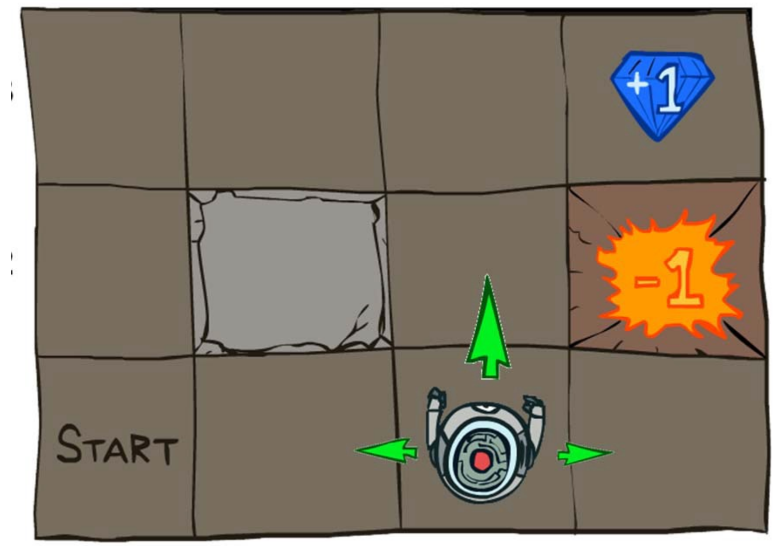
CS440/ECE448 Lecture 30: Markov Decision Processes

Mark Hasegawa-Johnson, 4/2023



Grid World

Invented and drawn by Peter Abbeel and Dan Klein, UC Berkeley CS 188

Outline

- Problem statement
- Utility
- The discount factor
- Value Iteration
- Policy Iteration

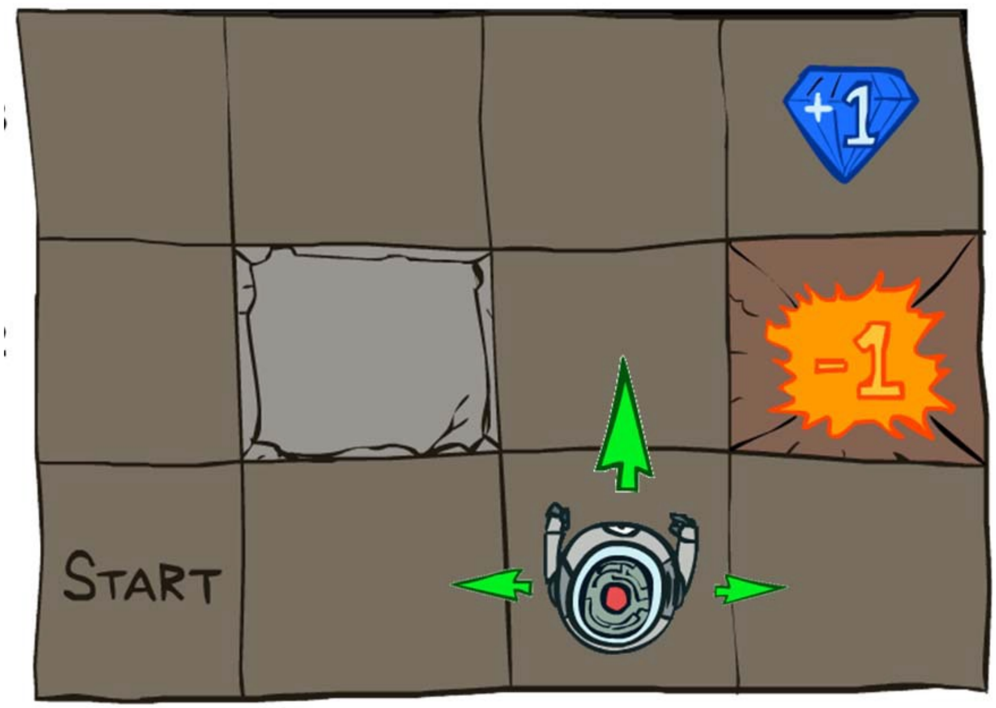
How does an intelligent agent plan its actions?

- If there is no randomness: Use A* search to plan the best path
- If there is an adversary: Use alpha-beta search to find the best path
- If our measurements are affected by random noise: Use Kalman filter to get a better estimate of current position
- What if our movements are affected by randomness?

Example: Grid World

Invented by Peter Abbeel and Dan Klein

- Maze-solving problem: state is $s = (i, j)$, where $0 \leq i \leq 2$ is the row and $0 \leq j \leq 3$ is the column.
- The robot is trying to find its way to the diamond.
- If it reaches the diamond, it gets a reward of $R((0,3)) = +1$ and the game ends.
- If it falls in the fire it gets a reward of $R((1,3)) = -1$ and the game ends.

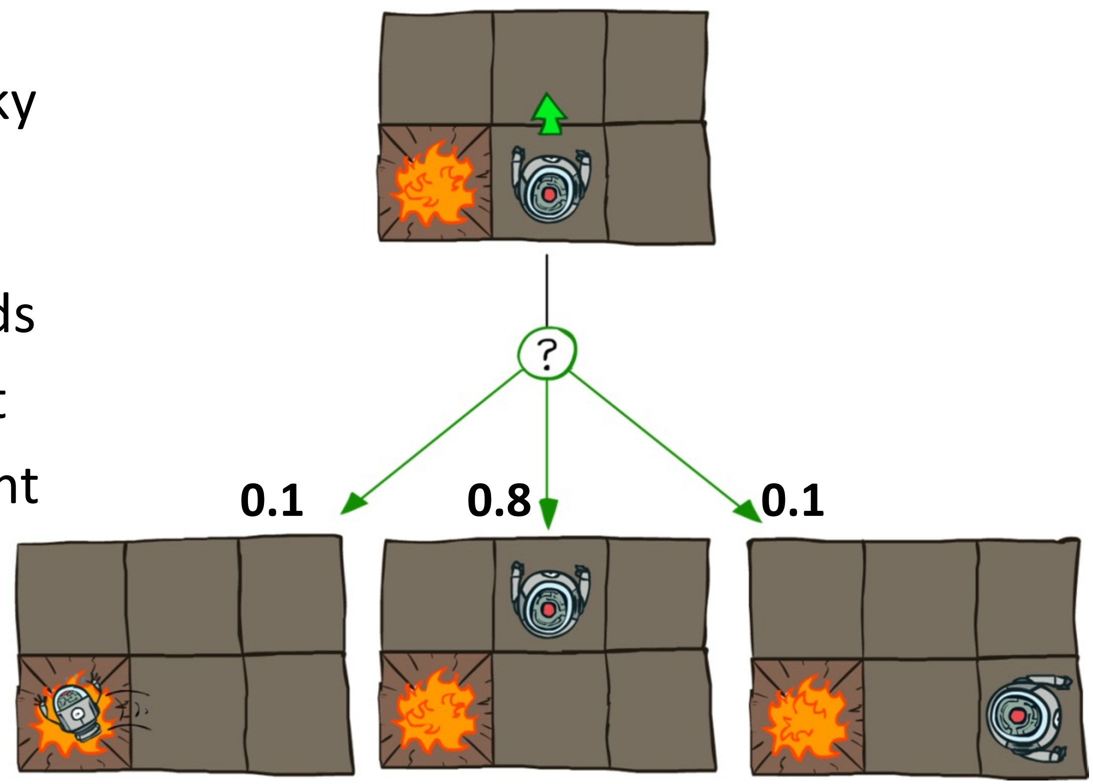


Example: Grid World

Invented by Peter Abbeel and Dan Klein

Randomness: the robot has shaky actuators. If it tries to move forward,

- With probability 0.8, it succeeds
- With probability 0.1, it falls left
- With probability 0.1, it falls right



Markov Decision Process

A Markov Decision Process (MDP) is defined by:

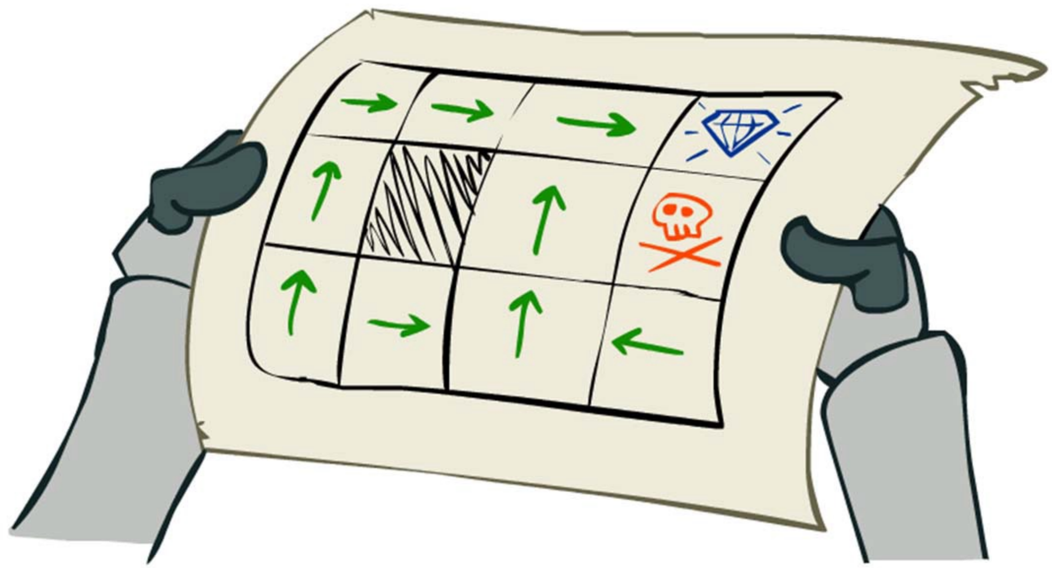
- A set of states, $s \in \mathcal{S}$
- A set of actions, $a \in \mathcal{A}$
- A transition model, $P(S_{t+1} = s_{t+1} | S_t = s_t, A_t = a_t)$
  - S_t is the state at time t
  - A_t is the action taken at time t
- A reward function, $R(s)$

Solving an MDP: The Policy

- The solution to a maze is a path: the shortest path from start to goal
- In MDP, finding 1 path is not enough: randomness might cause us to accidentally deviate from the optimal path.

Solving an MDP: The Policy

- Since P and R depend only on the state (the model is Markov), a complete solution can be expressed as follows:
- What is the best action to take in any given state?
- A policy, $a = \pi(s)$, is a function telling you, for any state s, what is the best action to take in that state.



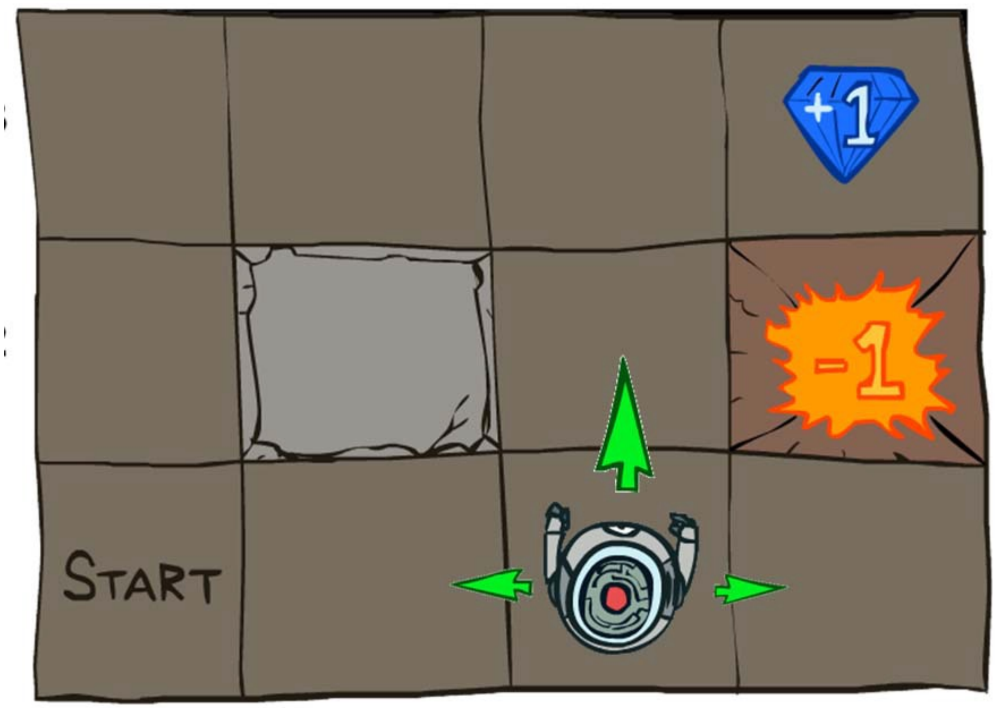
Outline

- Problem statement
- Utility
- The discount factor
- Value Iteration
- Policy Iteration

Utility

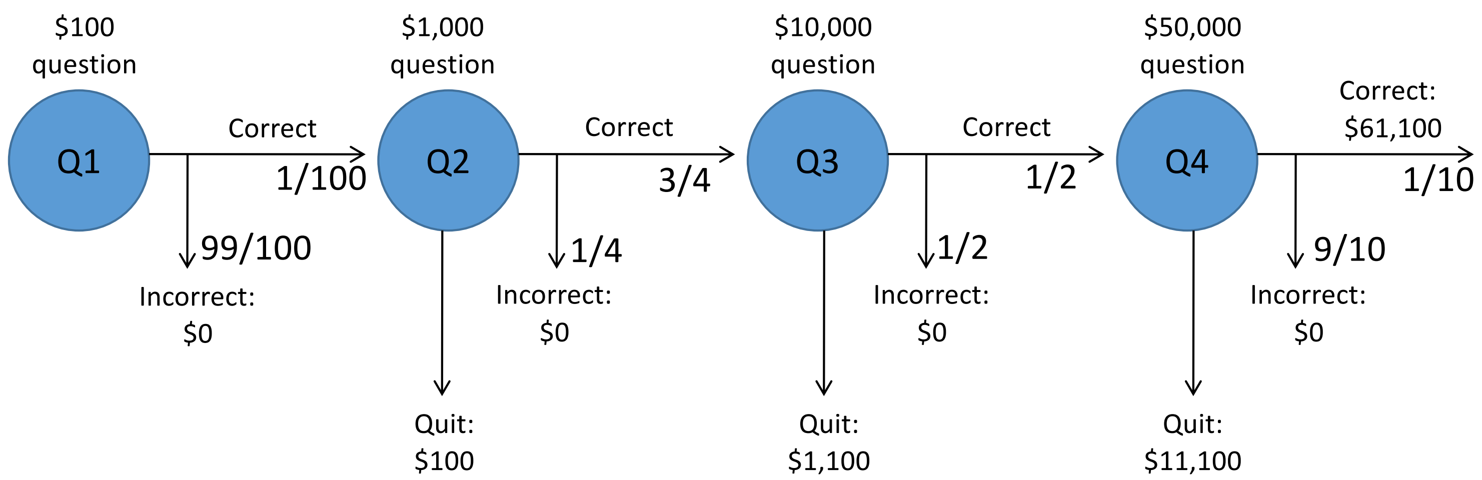
The utility of a state, U(s), is defined to be:

- the sum of all current and future rewards that can be achieved if we start in state s,
- …if we choose the best possible sequence of actions,
- …and if we average over all possible results of those actions.



Example: Game show

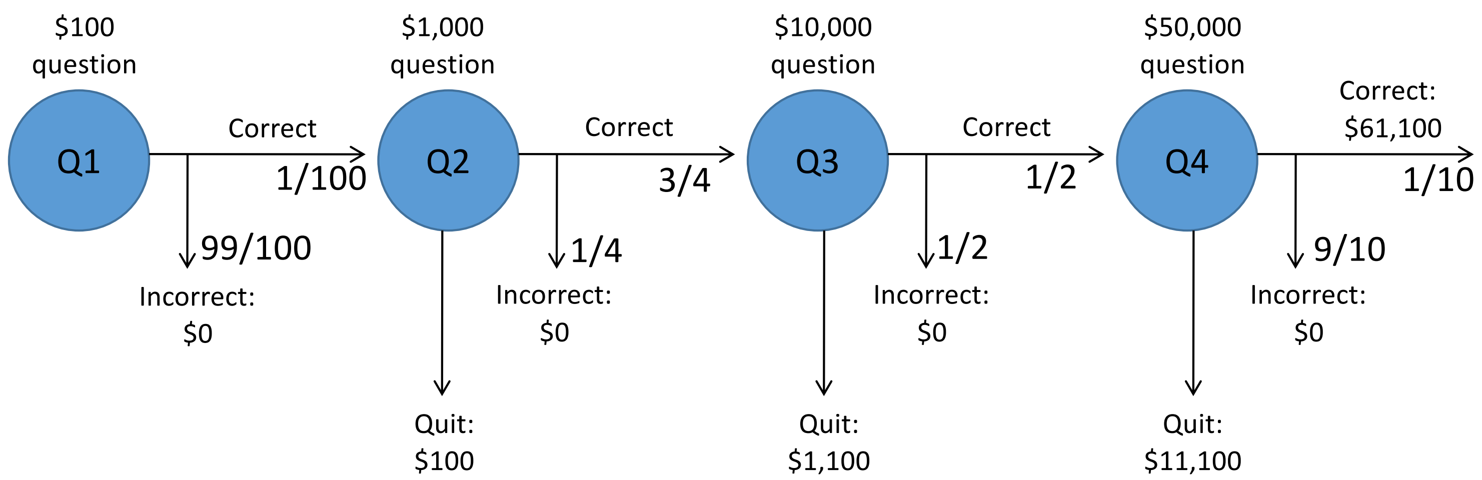
- You've been offered a spot as a contestant in a game show.
- Reward: you receive successively larger prizes for each question you answer correctly, but if you answer any question incorrectly, you lose it all.
- Transition: the questions become harder and harder to answer.
- Actions: after each question, you can decide whether to take another question, or stop.



Example: Game show

Policy:

- If you've correctly answered N-1 questions, should you attempt question QN, or stop?

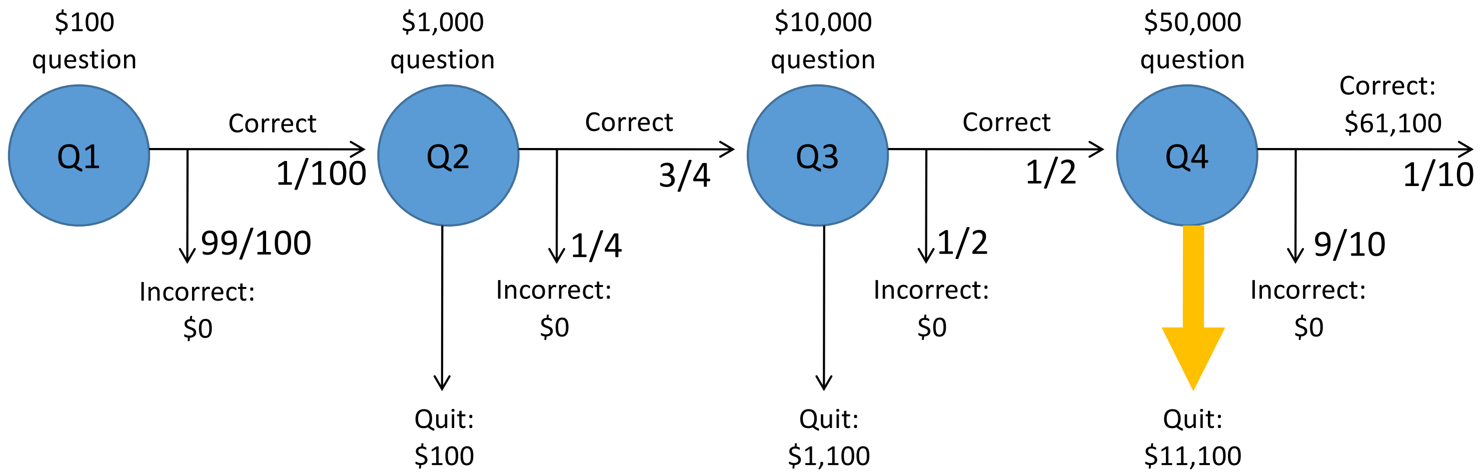


Example: Game show

Policy $\pi(Q4)$: If you've correctly answered 3 questions, should you attempt question Q4, or stop?

- If you stop: total reward is $11,100
- If you attempt Q4: expected total reward is $\frac{1}{10} \times 61100 + \frac{9}{10} \times 0 = \$6110$

Policy: $\pi(Q4) = \text{stop}$. Utility: $U(Q4) = \$11{,}100$

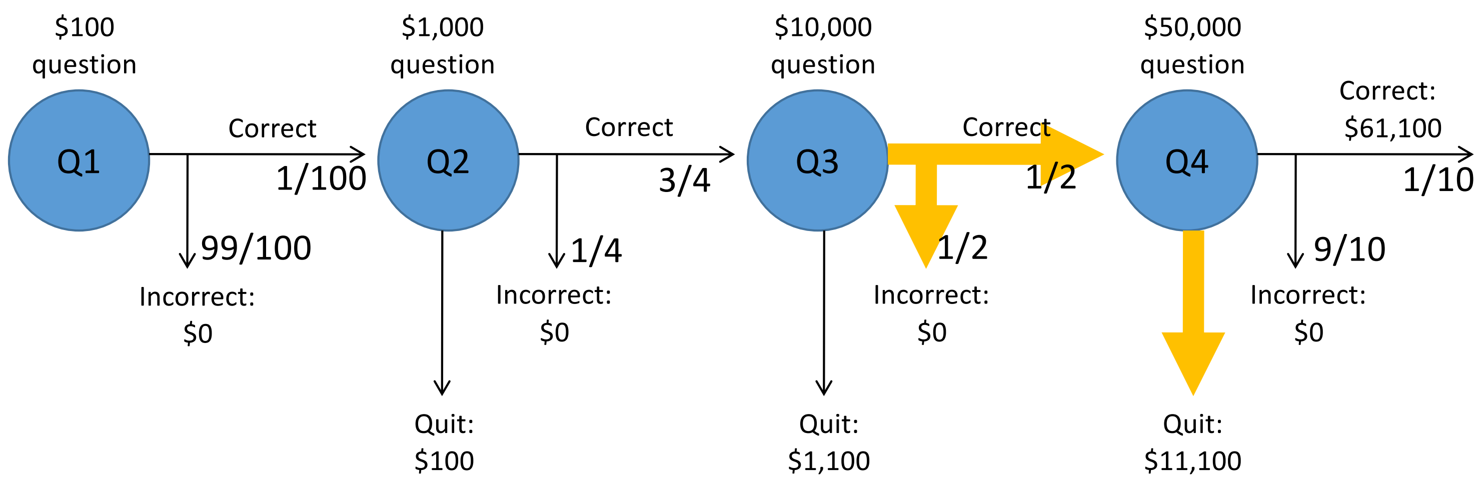


Example: Game show

Policy $\pi(Q3)$: If you've correctly answered 2 questions, should you attempt question Q3, or stop?

- If you stop: total reward is \$1,100
- If you attempt Q3: expected total reward is $\frac{1}{2} \times \$11{,}100 + \frac{1}{2} \times 0 = \$5550$

Policy: $\pi(Q3) = \text{continue.}$ Utility: $U(Q3) = \$5550$

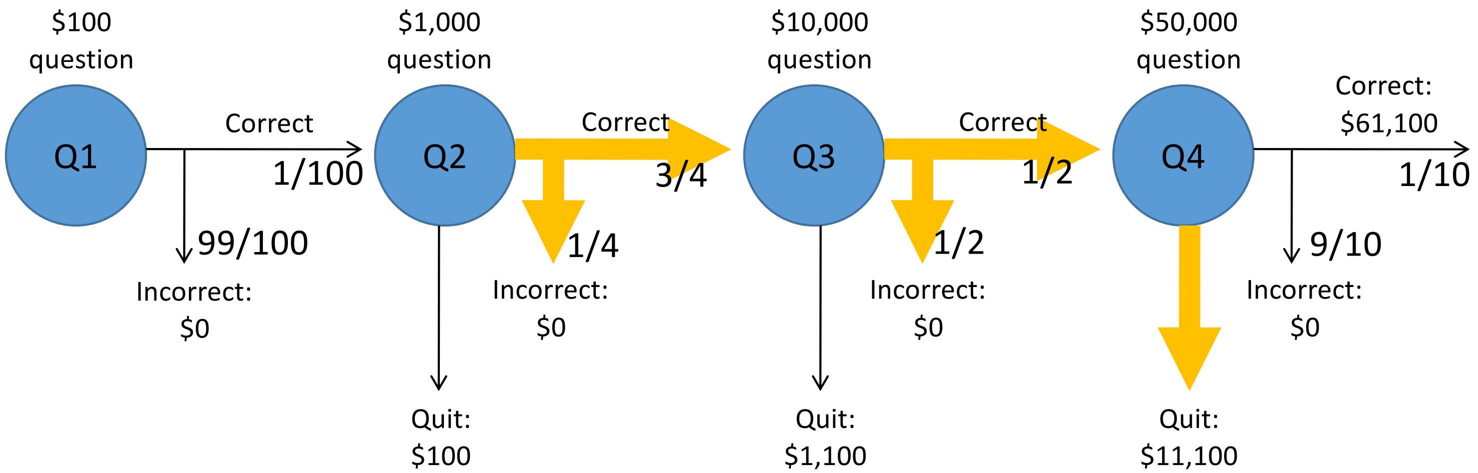


Example: Game show

Policy $\pi(Q2)$: If you've correctly answered 1 question, should you attempt question Q2, or stop?

- If you stop: total reward is \$100
- If you attempt Q2: expected total reward is $\frac{3}{4} \times \$5550 + \frac{1}{4} \times 0 = \$4162.50$

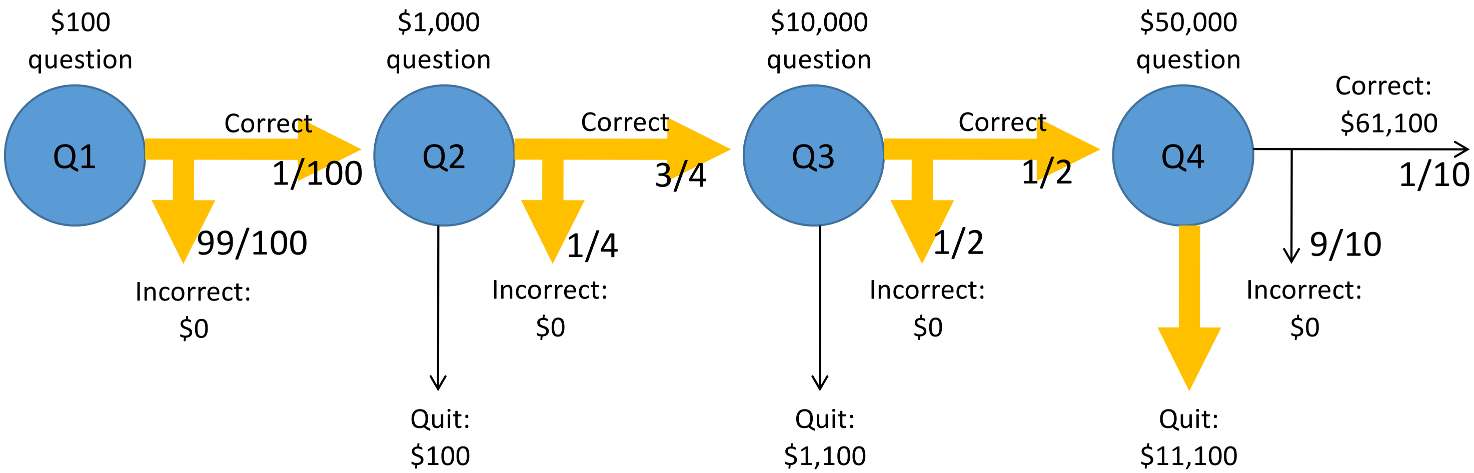
Policy: $\pi(Q2) = \text{continue.}$ Utility: $U(Q2) = \$4162.50$



Example: Game show

Policy $\pi(Q1)$: If you've correctly answered no questions, then you have nothing to lose, so even though the chance of success is very small, you might as well try it!

Policy: $\pi(Q1) = \text{continue.}$ Utility: $U(Q1) = \$41.63$



Utility

The utility of a state, U(s), is

- …the maximum, over all possible sequences of actions, of
- …the expected value, over all possible results of those actions, of
- …the total of all future rewards.

$$U(s_0) =$$

$$R(s_0) + \max_{a_0} \sum_{s_1} P(s_1|s_0, a_0) \left(R(s_1) + \max_{a_1} \sum_{s_2} P(s_2|s_1, a_1)(R(s_2) + \cdots) \right)$$

Utility

The utility of a state, U(s), is

- …the maximum, over all possible sequences of actions, of
- …the expected value, over all possible results of those actions, of
- …the utility of the resulting state.

$$U(s_0) = R(s_0) + \max_{a_0} \sum_{s_1} P(s_1 | s_0, a_0) U(s_1)$$

Outline

- Problem statement
- Utility
- The discount factor
- Value Iteration
- Policy Iteration

Discount factor

You have just won a contest sponsored by the Galaxia Foundation. They offer you the choice of two options:

- $60,000 right now, or…
- $1000 per year, paid to you and your heirs annually forever.

Which option is better?

Discount factor

- Inflation has averaged 3.8% annually from 1960 to 2021.
- Equivalently, \$1000 received one year from now is worth approximately \$962 today.
- A reward of \$1000 annually forever (starting today, t=0) is equivalent to an immediate reward of

$$R = \sum_{t=0}^{\infty} 1000(0.962)^t = \frac{1000}{1 - 0.962} = \$26{,}316$$

We call the factor $\gamma = 0.962$ the discount factor.

Discount factor

Why is a dollar tomorrow worth less than a dollar today?

- A dollar will buy less tomorrow
- The person paying you might go out of business
- You might have to move to California hence you wouldn't be able to collect

The discount factor, γ , is our model of the unknowable uncertainty of promised future rewards.

Public domain image of J. Wellington Wimpy, the character who popularized the saying "I will gladly pay you Tuesday for a hamburger today."

https://commons.wikimedia.org/wiki/File:Wimpyhotdog.png

The Bellman Equation

$$U(s_0) = R(s_0) + \gamma \max_{a_0} \sum_{s_1} P(s_1|s_0, a_0)U(s_1)$$

- The Bellman equation specifies the utility of the current state.
- In solving the Bellman equation, we also find the optimum action, which is the policy.
- However…

The Bellman Equation

$$U(s_0) = R(s_0) + \gamma \max_{a_0} \sum_{s_1} P(s_1|s_0, a_0)U(s_1)$$

- The Bellman equation is N nonlinear equations in N unknowns
- N is the number of states
- U(s) are the unknowns
- There is no closed-form solution; we must use an iterative solution

Outline

- Problem statement
- Utility
- The discount factor
- Value Iteration
- Policy Iteration

Value iteration

The Bellman Equation:

$$U(s) = R(s) + \gamma \max_a \sum_{s'} P(s'|s,a) U(s')$$

Value iteration solves the Bellman equation iteratively. In iteration number i, for $i = 0,1,\ldots,$

- For all states s, $U_i(s)$ is an estimate of $U(s)$
- Start out with $U_0(s) = 0$ for all states
- In the i^{th} iteration,

$$U_i(s) = R(s) + \gamma \max_a \sum_{s'} P(s'|s,a) U_{i-1}(s')$$

Quiz

Try the quiz!

https://us.prairielearn.com/pl/course_instance/129874/assessment/2340278

Ui-1(0)=0.1, Ui-1(1)=-0.5, R(0)=R(1)=1, gamma=0.6

Ui(0)=1+(0.6)max((0.6)(0.1)+(0.4)(-0.5),(0.8)(0.1)+(0.2)(-0.5))

Ui(1)=1+(0.6)max(,)

Pi(0)=1

Outline

- Problem statement
- Utility
- The discount factor
- Value Iteration
- Policy Iteration

Method 2: Policy Iteration

- Start with some initial policy π_0 and alternate between the following steps:
  - **Policy Evaluation:** calculate the utility of every state under the assumption that the given policy is fixed and unchanging, i.e, $U^{\pi}(s)$
  - **Policy Improvement:** calculate a new policy π_{i+1} based on the updated utilities.

- Notice it's kind of like gradient descent in neural networks:
  - Policy evaluation: Find ways in which the current policy is suboptimal
  - Policy improvement: Fix those problems

- Unlike Value Iteration, this is guaranteed to converge in a finite number of steps, as long as the state space and action set are both finite.

Step 1: Policy Evaluation

Policy Evaluation: Given a fixed policy π, calculate the **<u>policy-dependent utility</u>**, $U^{\pi}(s)$, for every state *s*

$$\boxed{U^{\pi}(s) = R(s) + \gamma \sum_{s'} P(s' \mid s, \pi(s)) U^{\pi}(s')}$$

Notice how this differs from the Bellman equation:

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

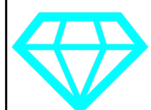
The difference is that policy evaluation is N _linear_ equations in N unknowns, whereas the Bellman equation is N _nonlinear_ equations in N unknowns (N=# states).

Example: Grid World

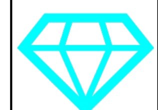
Policy Evaluation: $$U^{\pi^0}(s) = R(s) + \gamma \sum_{s'} P(s'|s,a) U^{\pi^0}(s')$$

- Assume a "loitering penalty" of R(s)=-0.04 for all non-terminal states
- Assume the initial policy is $\pi^0(s) = \text{Right}$ for all states
- Solve the linear equations to find $U^{\pi^0}(s)$ for all states

$U^{\pi^0}(s)$

$+0.50$	$+0.69$	$+0.74$	
-0.65		-0.90	
-1.40	-1.44	-1.39	-1.40

$\pi^0(s)$

$\rightarrow$	$\rightarrow$	$\rightarrow$	
$\rightarrow$		$\rightarrow$	
$\rightarrow$	$\rightarrow$	$\rightarrow$	$\rightarrow$

Step 2: Policy Improvement

- **Policy Evaluation**: Given a fixed policy π, calculate the **policy-dependent utility**, $U^{\pi}(s)$, for every state s

$$\boxed{U^{\pi}(s) = R(s) + \gamma \sum_{s'} P(s' \mid s, \pi(s)) U^{\pi}(s')}$$

- **Policy Improvement**: Given $U^{\pi}(s)$ for every state s, find an improved $\pi(s)$

$$\boxed{\pi^{i+1}(s) = \arg\max_{a \in A(s)} \sum_{s'} P(s' \mid s, a) U^{\pi_i}(s')}$$

Policy Improvement: Iteration 1

Policy Evaluation: $U^{\pi_0}(s) = R(s) + \gamma \sum_{s'} P(s'|s, \pi_0(s)) U^{\pi_0}(s')$

Policy Improvement: $\pi_1(s) = \underset{a}{\operatorname{argmax}} \sum_{s'} P(s'|s, a) U^{\pi_0}(s')$

$\pi_1(s)$

$\rightarrow$	$\rightarrow$	$\rightarrow$	
$\uparrow$		$\uparrow$	
$\uparrow$	$\rightarrow$	$\uparrow$	$\uparrow$

$U^{\pi_0}(s)$

$+0.50$	$+0.69$	$+0.74$	
-0.65		-0.90	
-1.40	-1.44	-1.39	-1.40

$\pi_0(s)$

$\rightarrow$	$\rightarrow$	$\rightarrow$	
$\rightarrow$		$\rightarrow$	
$\rightarrow$	$\rightarrow$	$\rightarrow$	$\rightarrow$

Summary

- MDP defined by states, actions, transition model, reward function
- The "solution" to an MDP is the policy: what do you do when you're in any given state
- The Bellman equation tells the utility of any given state, and incidentally, also tells you the optimum policy. The Bellman equation is N nonlinear equations in N unknowns (the policy), therefore it can't be solved in closed form.
- Value iteration:
  - At the beginning of the (i+1)'st iteration, each state's value is based on looking ahead i steps in time
  - … so finding the best action = optimize based on (i+1)-step lookahead
- Policy iteration:
  - Find the utilities that result from the current policy,
  - Improve the current policy