RUTGERS

THE STATE UNIVERSITY
OF NEW JERSEY

Markov Decision Process

(MDP)

Edgar Granados

Sequential Decision Making

Definition

- Episodic Decisions...
- Continuous decision making?

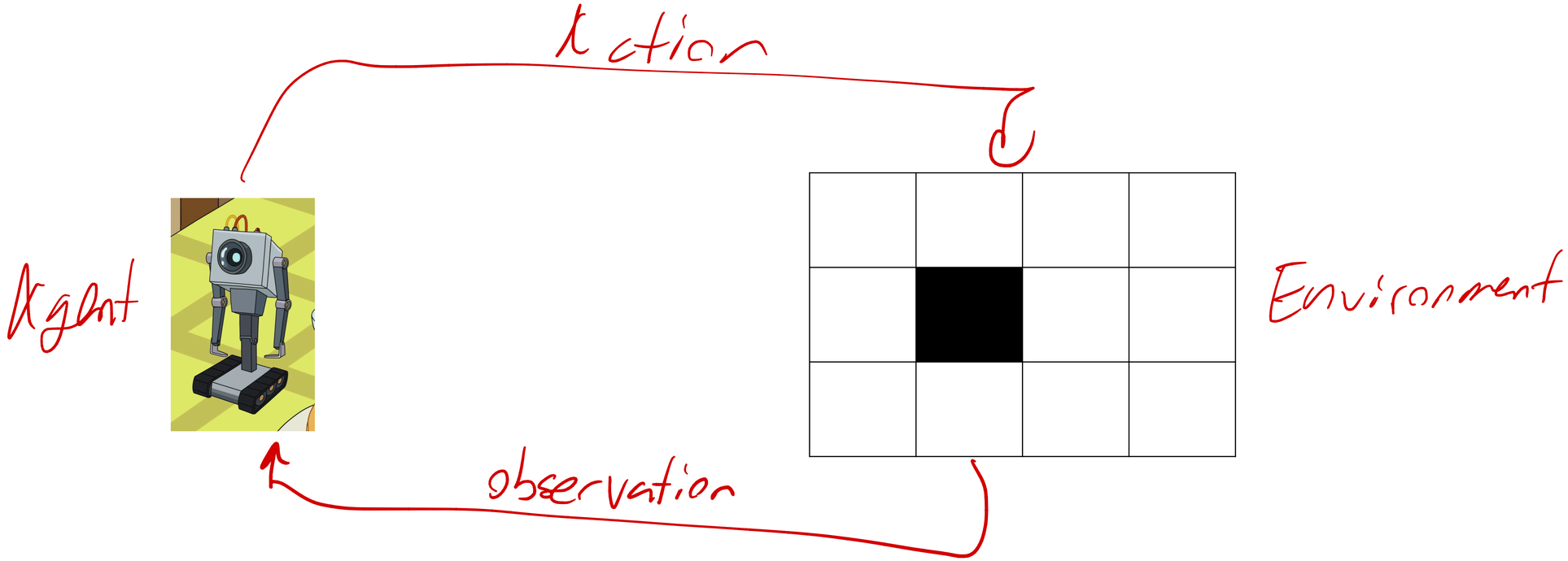
Sequential Decision Making

Definition

- Episodic Decisions...
- Continuous decision making?

- **Take actions that affects later decisions**

Agent-Environment Interaction



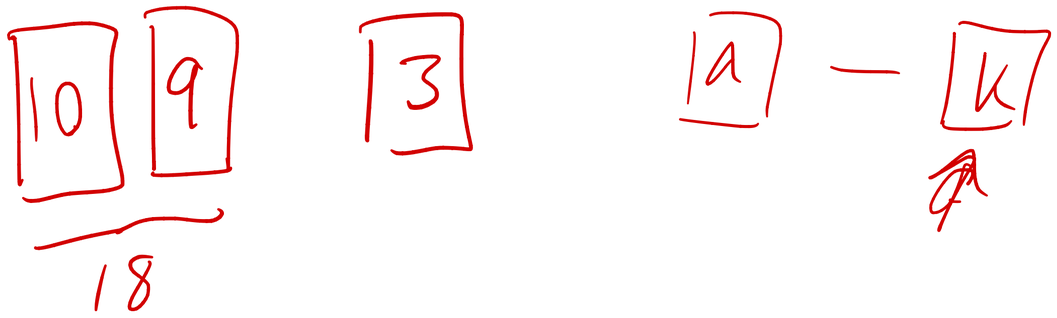
- Probabilistic actions
- Deterministic observations

- Sequence that yields the best payout - Reward

Agent-Environment Interaction

Example

- Blackjack



Markov Decision Process

Definition

- Tuple (S, A, T, R):
  - S - States
  - A - Actions
  - T - Transition Matrix
  - R - Reward function

Policy $\pi^*: S \rightarrow A$

MDPs

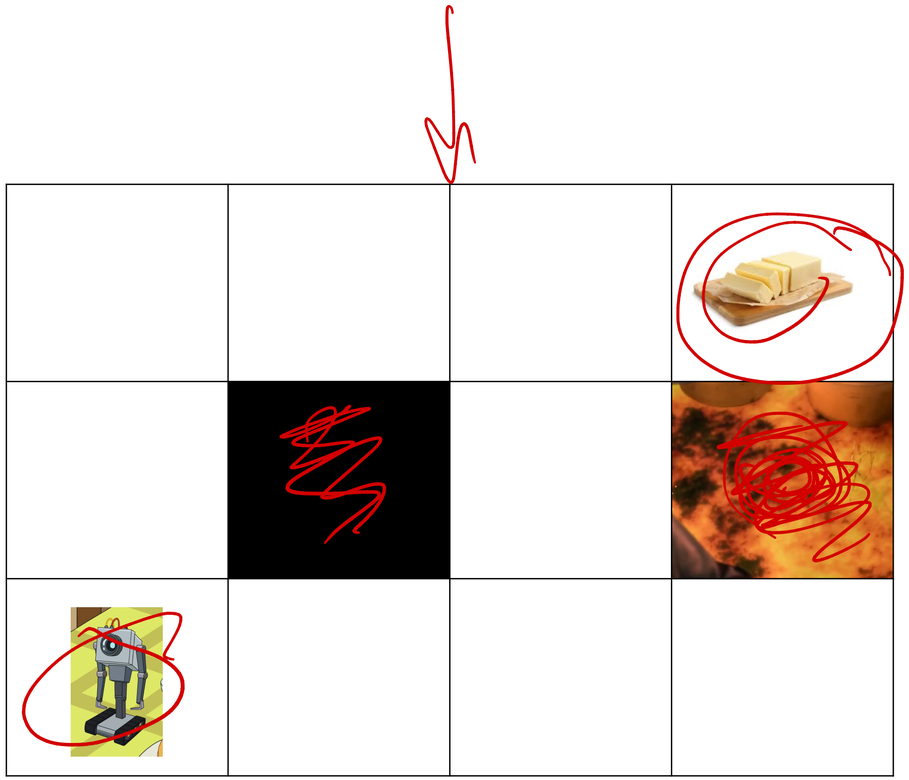
Markov?

- Markov Assumption!
- Returns a **Policy**

MDP

States

- States - Representation of all reveal information for predicting future states
  - Describes the configuration of the system at a given moment

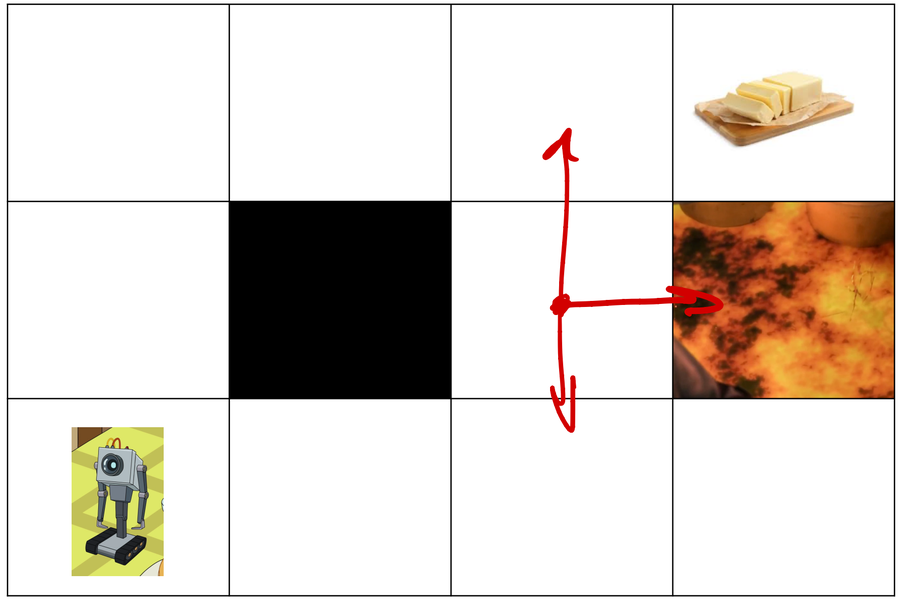


MDP

Actions

- When executed, actions modify states
- Determine the best **sequence of actions**

$A = \{L, R, U, D\}$



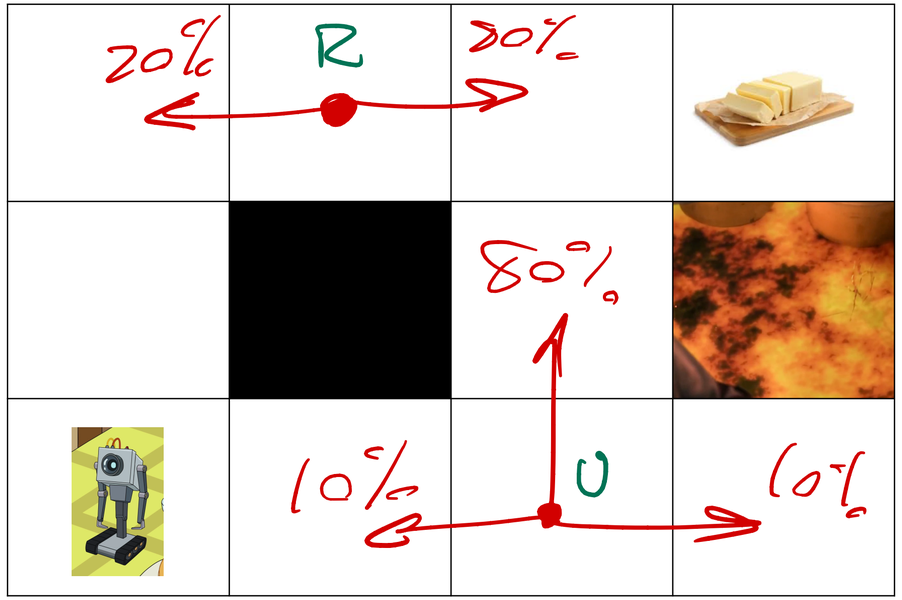
MDP

Transition Function

- $T(s_t, a_t, s_{t+1}) = P(s_{t+1} \mid s_t, a_t)$

80% → correct state | action

20% → incorrect state(s)

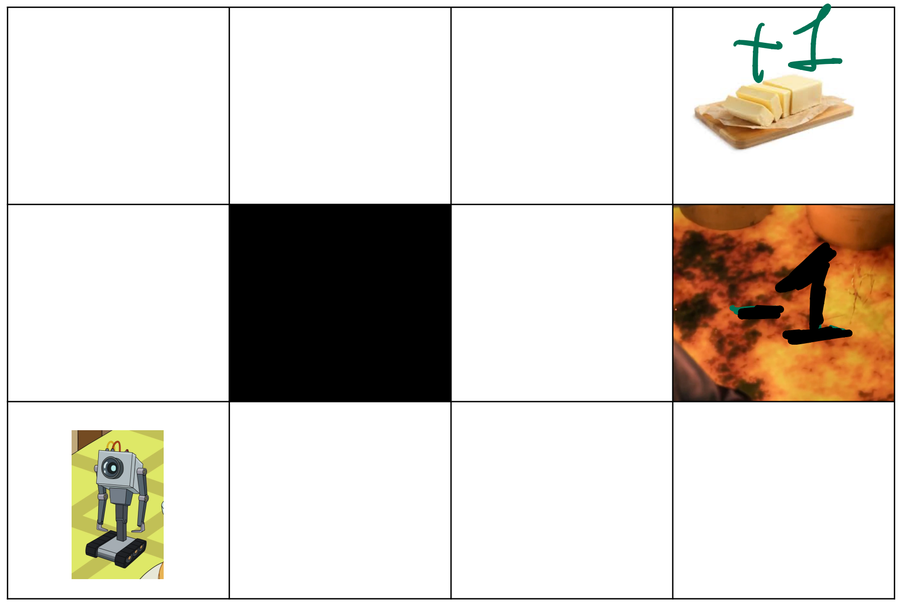


MDP

Reward Function

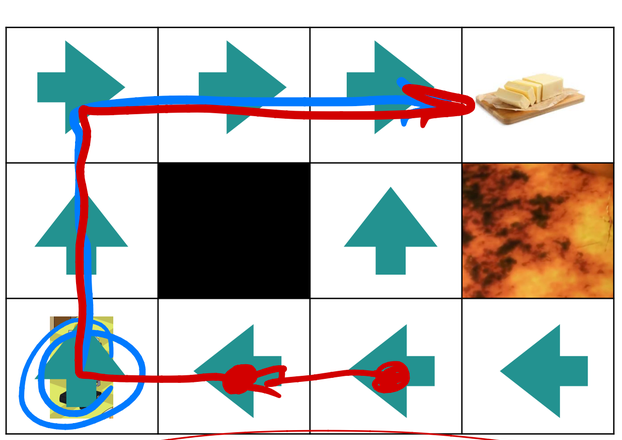
- $R(s_t, a_t)$
- Direct agent towards desirable states & away from unwanted ones
- Maximize the cumulative reward

- MDP: Model complex concurrent tasks by simply assigning rewards to the states



Rewards

Appropriate rewards?

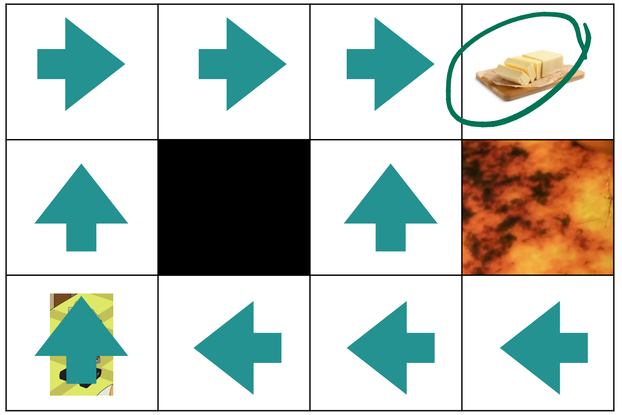


$$R(s_t, move) = -0.04$$

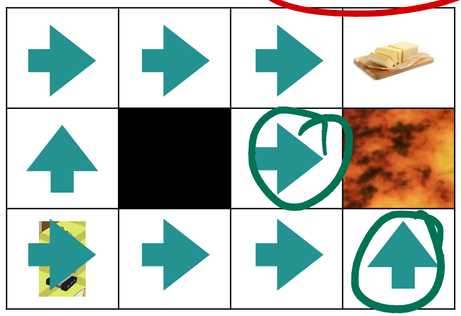
Rewards

Appropriate rewards?

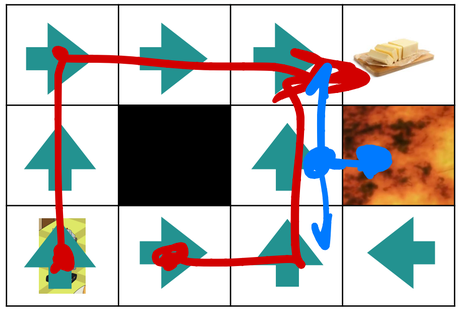
$R(s_t, move) = -0.04$



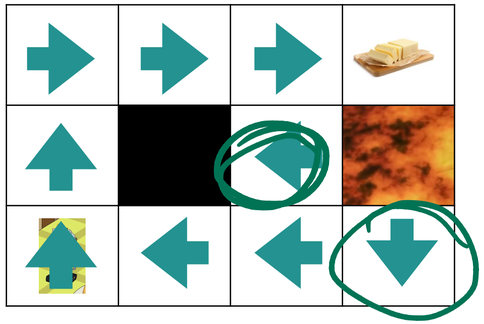
$R(s_t, move) < -1.6284$



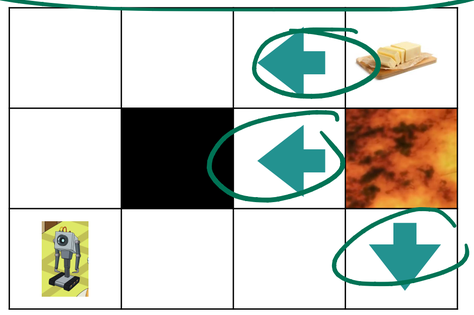
$-0.4278 < R(s_t, move) < -0.085$



$-0.0221 < R(s_t, move) < 0$



$R(s_t, move) > 0$



MDP

- Overall utility:
  - $V[s_0, s_1, \ldots] =$ $R(s_0) + R(s_1) + (\ldots)$

MDP

- Overall utility using discounts: $0 < \gamma < 1$
  - $V[s_0, s_1, \ldots] = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \gamma^3 R(s_3) + (\ldots)$

$\frac{R_{max}}{1-\gamma}$ ⟼ Max Possible total Reward

Bellman's Equation

- $\pi : S \rightarrow A$:
  - $V^{\pi}(s) = \sum_{t=0}^{\infty} \gamma^t E[R(s_t) \mid s_0 = s)] = R(s) + \sum_{t=1}^{\infty} \gamma^t E[R(s_t) \mid s_0 = s]$

$$= R(s) + \gamma \sum_{t=0}^{\infty} \gamma^t E[R(s) \mid s_0 \in T(s, \pi(s))]$$

$$= R(s) + \gamma \sum_{s' \in T(s,\pi(s))} P(s' \mid s, \pi(s)) \sum_{t=0}^{\infty} \gamma^t E[R(s_t) \mid s_0 = s]$$

Bellman's Equation

$$V^{\pi}(s) = R(s, \pi(s)) + \gamma \sum_{s' \in S} T(s, \pi(s), s') V^{\pi}(s')$$

Bellman's Equation

$$V^{\pi}(s) = R(s, \pi(s)) + \gamma \sum_{s' \in S} T(s, \pi(s), s') V^{\pi}(s')$$

- Dynamic programming:
  - Solve complex problems by breaking them down into simpler ones

Optimal Policy

- An optimal policy satisfies:
  - $\forall s \in S : \pi^* \in arg\,max V^{\pi}(s)$

Optimal Policy

- An optimal policy satisfies:
  - $\forall s \in S : \pi^* \in arg\, max V^{\pi}(s)$

- The necessary and sufficient condition:

$$V^{\pi}(s) = \max_{\pi}[R(s, \pi(s)) + \gamma \sum_{s' \in S} T(s, \pi(s), s') V^{\pi}(s')]$$

Solving MDPs

$$V_{k+1}(s) = R(s) + \gamma \max_{a \in A(S)} \left[\sum_{s' \in T(s,a)} P(s'|s,a) V_k(s') \right]$$

Value Iteration

1. $V_0 \equiv 0, k = 0$
2. Compute v_{k+1}, $\forall s \in S$
3. (Optional) if $|V_{k+1}(s) - V_k(s)| > \delta$:
    1. $|V_{k+1}(s) - V_k(s)| = \delta$
4. Repeat until $|V_{k+1}(s) - V_k(s)| < \epsilon \forall s \in S$

$\epsilon > 0$

$$N = \left\lceil \frac{\log\left[\frac{2R_{max}}{\epsilon(1-\gamma)}\right]}{\log(1-\gamma)} \right\rceil$$

$\uparrow \gamma \Rightarrow$ slow

Solving MDPs

Value Iteration

$$V_{k+1}(s) = R(s) + \gamma \max_{a \in A(S)} \left[\sum_{s' \in T(s,a)} P(s'|s,a) V_k(s') \right]$$

$\gamma = 1$

1. $V_0([4,3]) = 1, V_0([4,2]) = -1$

$V_0(s') = 0$

2. Start at [3,3] $\sum_{s' \in T(s,a)} P(s'|s,a) V_0(s')$
   1. Up: $.8(0) + .1(0) + .1(1) = .1$
   2. Right: $.8(1) + .1(0) + .1(0) = .8$
   3. Down: $.8(0) + .1(0) + .1(1) = .1$
   4. Left: $.8(0) + .1(0) + .1(0) = 0$

Max

$V_1[(3,3)] = -0.04 + .8 = .76$

Right

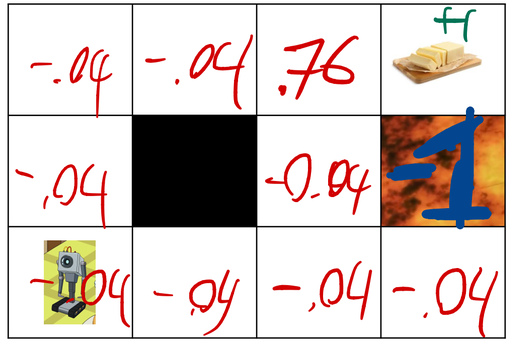
Solving MDPs

Value Iteration

$$V_{k+1}(s) = R(s) + \gamma \max_{a \in A(S)} \left[\sum_{s' \in T(s,a)} P(s'|s,a) V_k(s') \right]$$

1. $V_0([4,3]) = 1, V_0([4,2]) = -1$
2. Start at [3,3] $\Sigma_{s' \in T(s,a)} P(s'|s,a) V_0(s')$
   1. Best action is going **right**
3. All other $s \in S$, $V_1(s) = -0.04$
4. At [3,3], compute V_2
   1. Up $.8(.76) + .1(1) + .1(-.04) = .668$
   2. Right $.8(1) + .1(-.04) + .1(.76) = .872$
   3. Down $.8(-.04) + .1(1) + .1(-.04) = .064$
   4. Left $.8(-.04) + .1(-.04) + .1(.76) = .039$

$V_2[(3,3)] = -0.04 + .872 = .832$ Go Right



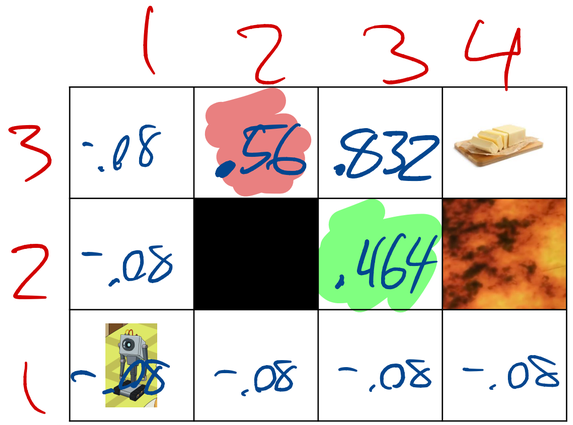
Solving MDPs

Value Iteration

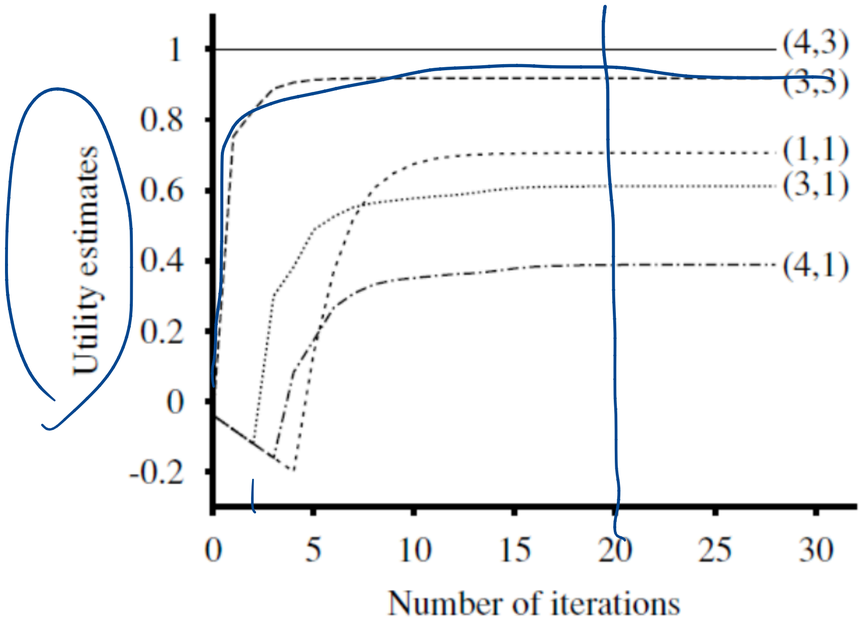
$$V_{k+1}(s) = R(s) + \gamma \max_{a \in A(S)} \left[\sum_{s' \in T(s,a)} P(s'|s,a) V_k(s') \right]$$

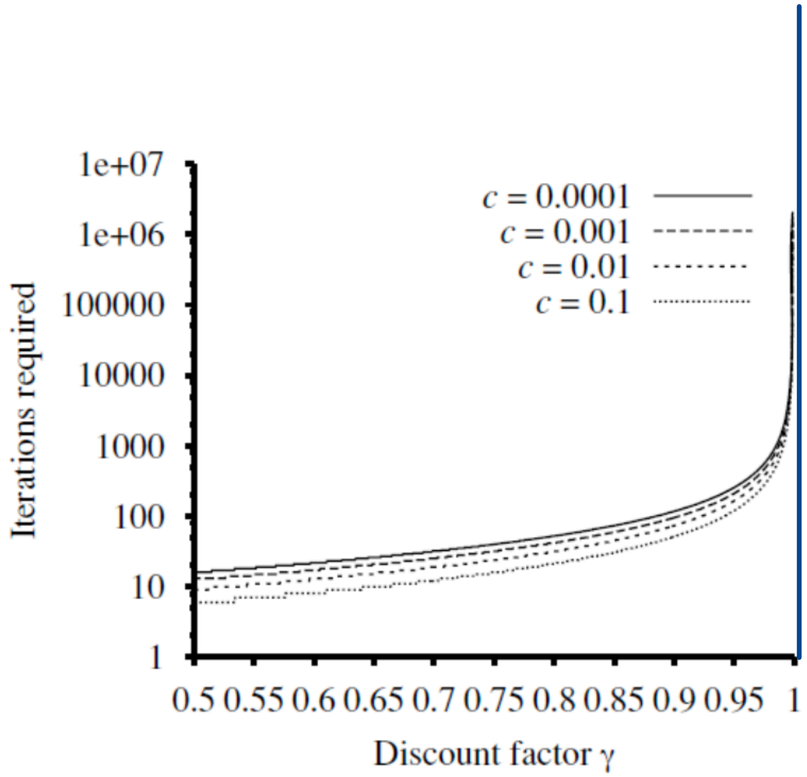
1. $V_0([4,3]) = 1, V_0([4,2]) = -1$
2. Start at [3,3]. $\sum_{s' \in T(s,a)} P(s'|s,a) V_0(s')$
    1. Best action is going **right**
3. All other $s \in S$, $V_1(s) = -0.04$
4. At [3,3], compute V_2
5. $V_2([2,3]) = -.04 + (.8(.76) + .2(-.04)) = .56$
6. $V_2([3,2]) = -.04 + (.8(.76) + .1(-.04) + .1(-1)) = .464$

$V_2 =$



Example





$\gamma = 1$

$\epsilon = c R_{max}$

Solving MDPs

Policy Iteration

- Start with some π_0
- Repeat until convergence:
  - Policy Evaluation $V_{k+1}(s) = R(s) + \gamma \sum_{s'} P(s'|s, \pi_k(s)) V_k(s')$
  - Policy Improvement $\pi_{k+1}(s) = \underset{a \in A(s)}{\text{argmax}} \sum_{s'} P(s'|s, a) V_{k+1}(s')$

$\gamma = 1$

$O(n^3)$

Solving MDPs

Policy Iteration

- Start with some π_0: All up
- Compute values:
  - $V_1([3,3]) =$
  - $V_1([3,2]) =$
  - $V_1([2,3]) =$
- Compute updated policy: $\sum_{s'} P(s'|s,a)V_{k+1}(s')$
  - For [3,3]:
    - Right
    - Up
    - Left
    - Down