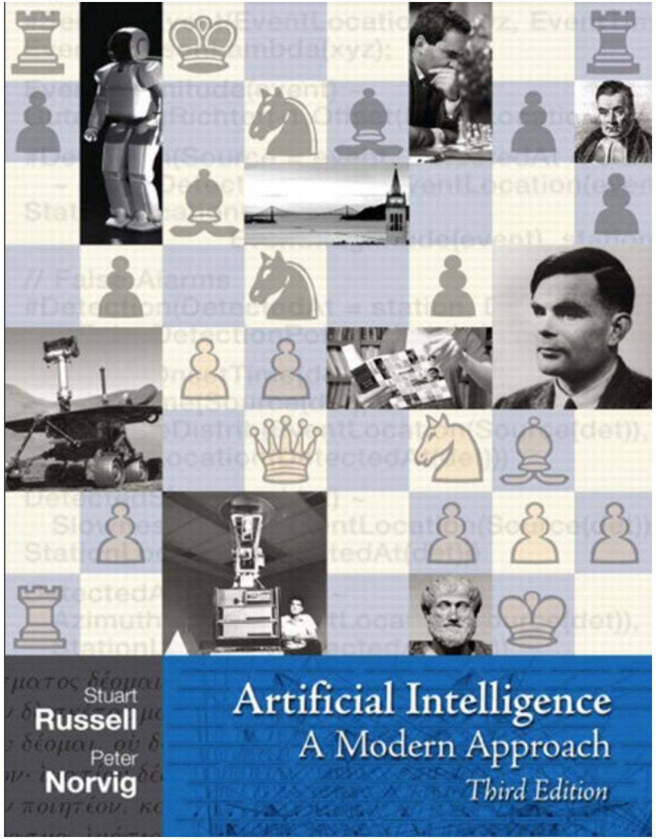
Markov Decision Process

Chao Lan

Reference

Section 17.1-17.3



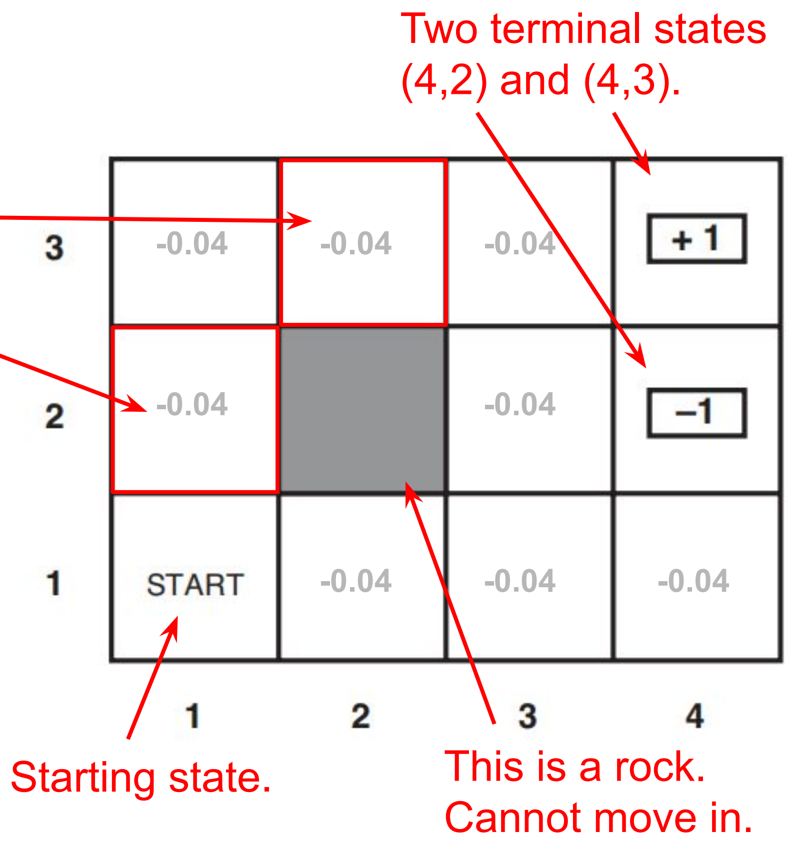
Problem Setting

Consider a grid world where each cell is a “state” described by its coordinate e.g., (1,2) and (2,3).

An agent can travel between states by taking an action from {up, down, left, right}.

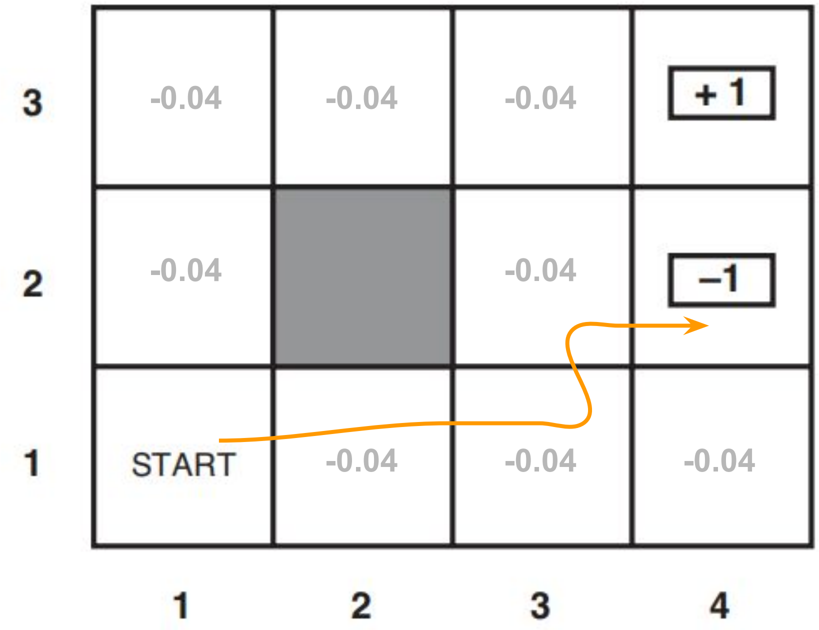
Agent stays at the current state if hitting wall/rock
- action “up” at (2,1) results in (2,1), not (2,2).
- action “left” or “right” at (1,2) results in (1,2).

Each state has a “reward” and the agent collects the reward when reaching the state (every time).



Example Travel and Reward Collection

Step	State(S)	Reward(R)	Action(A)
0	(1,1)	-0.04	right
1	(2,1)	-0.04	right
2	(3,1)	-0.04	up
3	(3,2)	-0.04	right
4	(4,2)	-1	stop
Total Reward		-1.16	



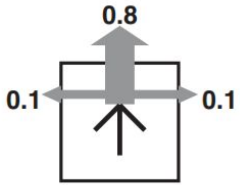
Stochastic Move

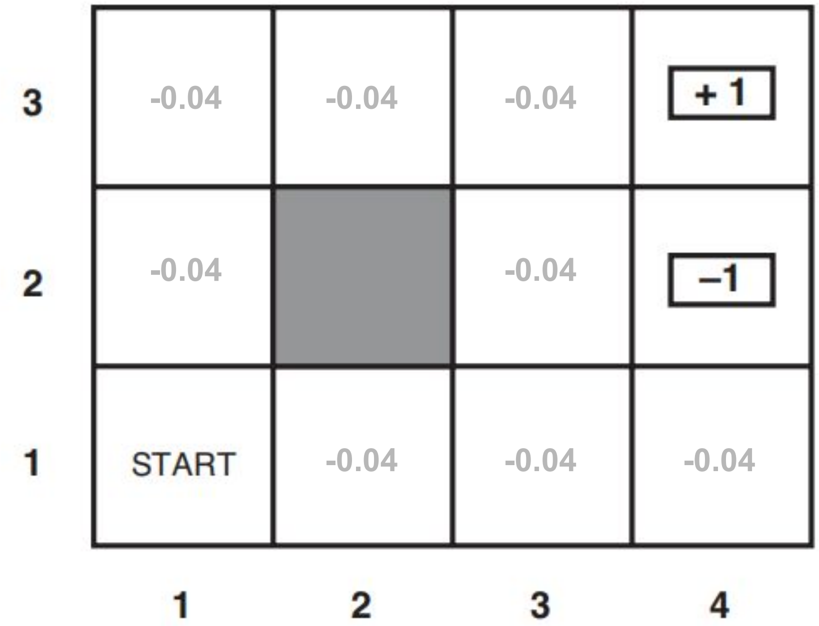
Suppose the result of an action is random.

For example, if we aim to move up

- 80% chance we will move up
- 10% chance we will move left
- 10% chance we will move right

Similar to action left/right/down.





Concept: Transition Probability

Characterize random results of an action using transition probability.

$$\Pr\{S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a\}$$

Interpretation: what is the probability that if we take action *a* at state *s*,
we end up at state *s'* and collect reward *r*?

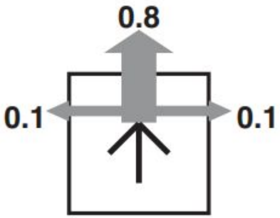
Example

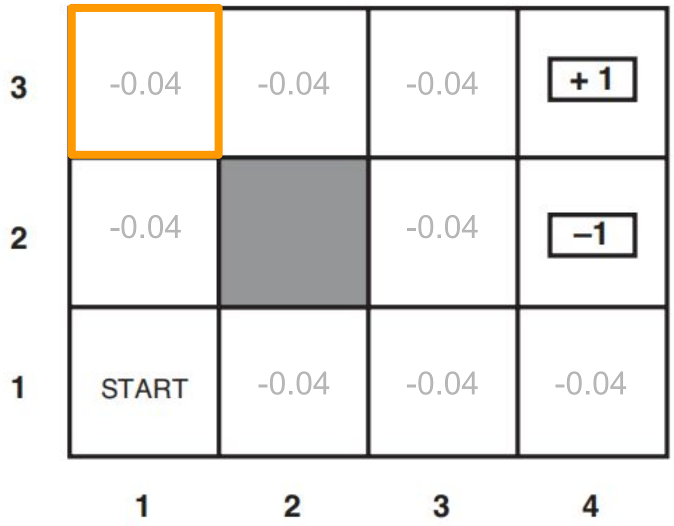
Pr{ (2,3), -0.04 | (1,3), right } = 0.8

Pr{ (1,3), -0.04 | (1,3), right } = 0.1

Pr{ (1,2), -0.04 | (1,3), right } = 0.1

Pr{ (1,2), -0.04 | (1,3), down } = 0.8





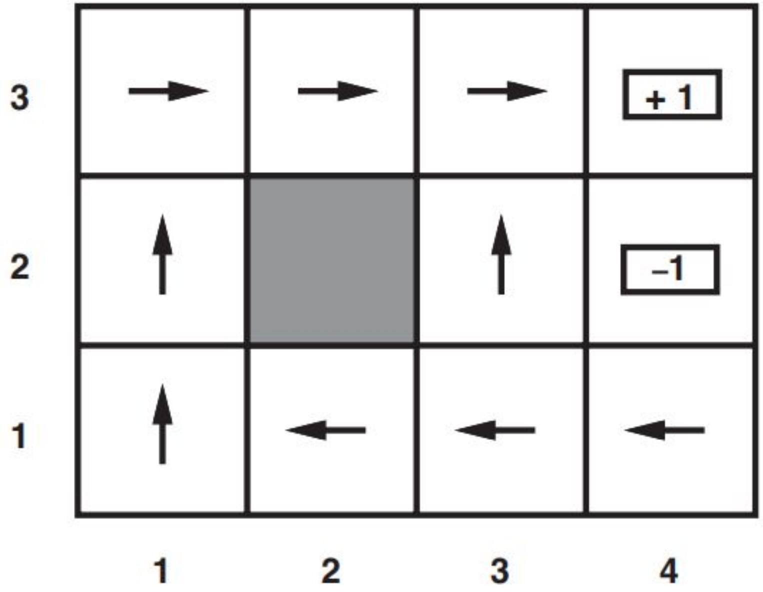
Concept: Policy

A policy π is a function mapping from a state s to its action π(s).

Example

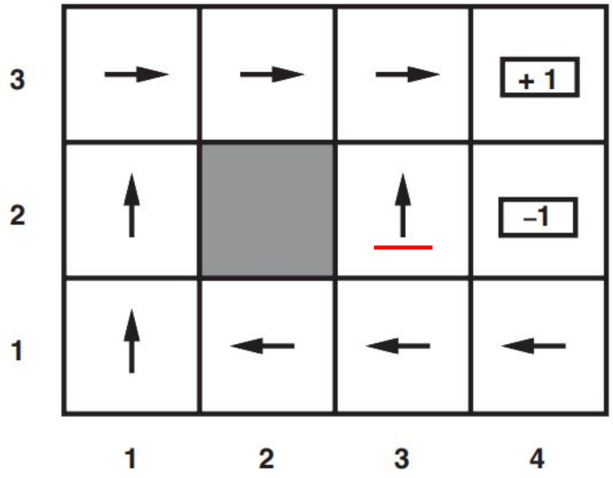
- π(1,1) = UP
- π(3,3) = RIGHT
- π(4,2) = LEFT

This set of actions forms one policy.

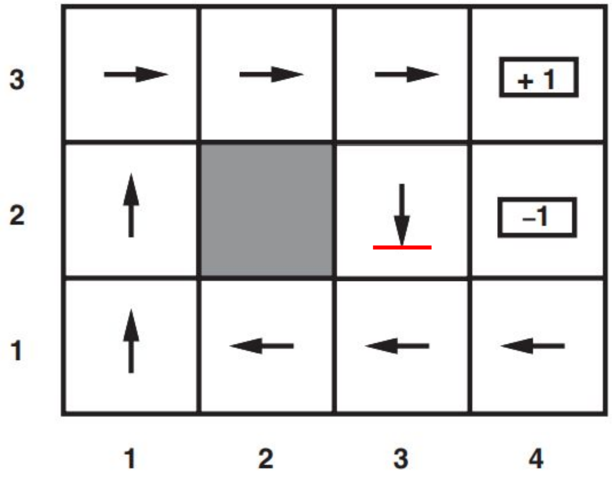


Examples of Different Policies

One Policy



Another Policy



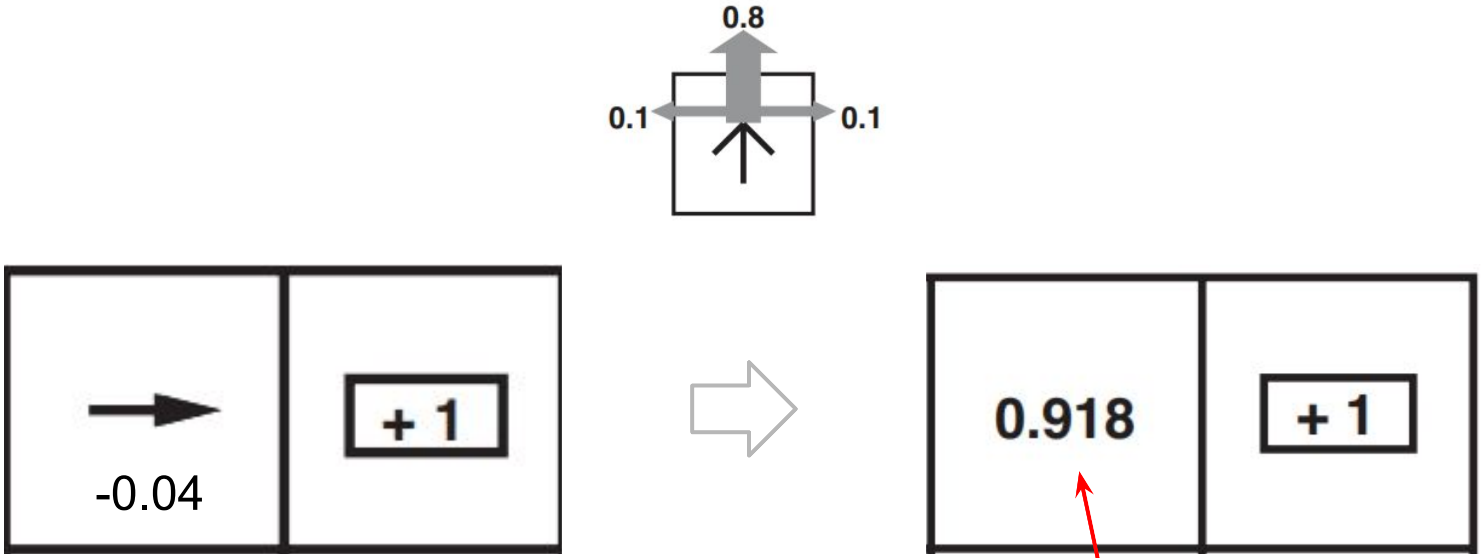
Concept: Expected Utility

The expected utility of policy π is the cumulative discounted rewards which can be collected by following π from any state s until termination.

$$U^{\pi}(s) = E\left[\sum_{t=0}^{\infty} \gamma^t R(S_t)\right]$$

where γ is the discount factor, chosen in [0,1].

Example



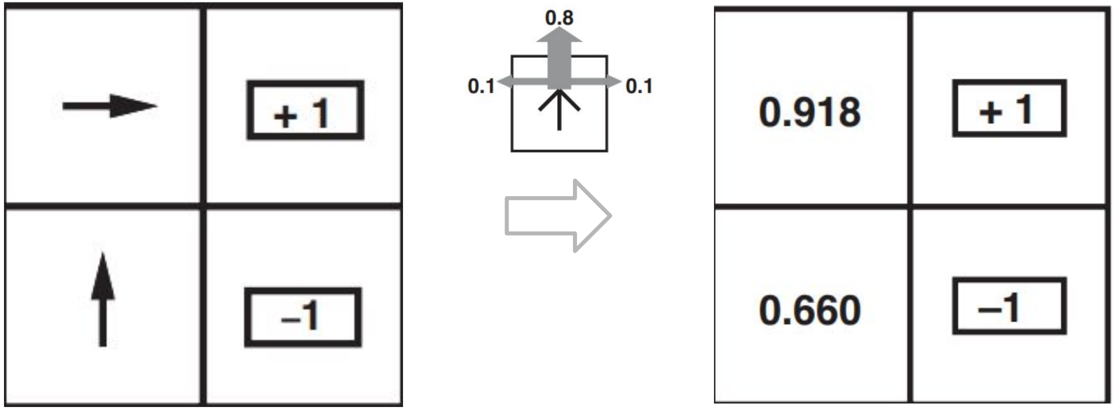
reward at this state is -0.04

expected utility of this state (based on the above transition probability and polity “right”) is 0.918

Optimal Policy

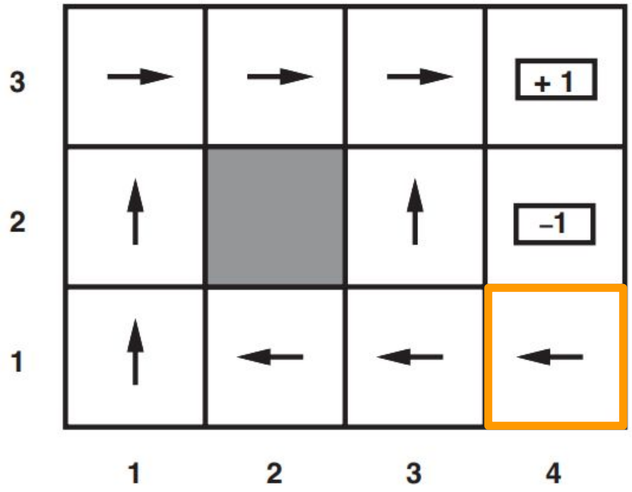
An optimal policy π* is one that maximizes the expected utility at every state.

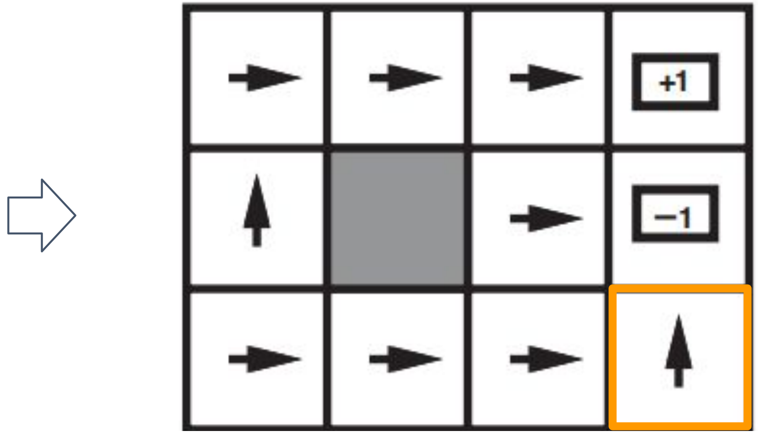
$$\pi^*_s = \underset{\pi}{\operatorname{argmax}}\, U^\pi(s)$$



Impact of Reward on Optimal Policy

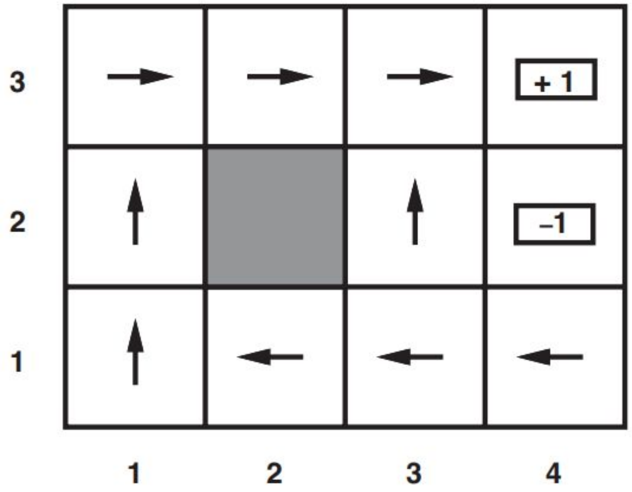
If we change reward from -0.04 to -2.

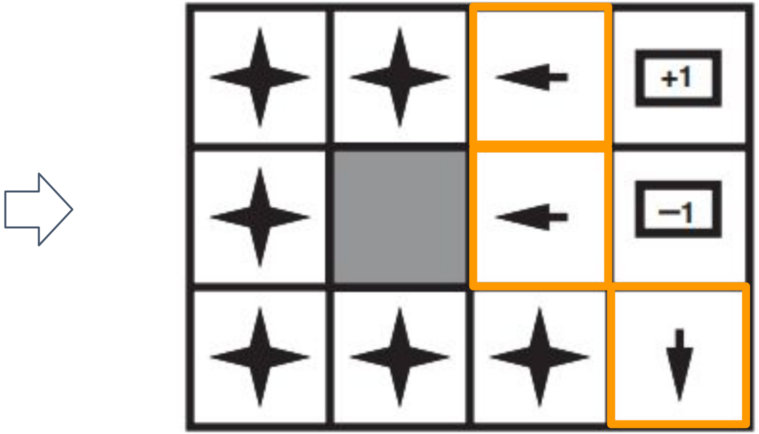




Impact of Reward on Optimal Policy

If we change reward from -0.04 to +10.





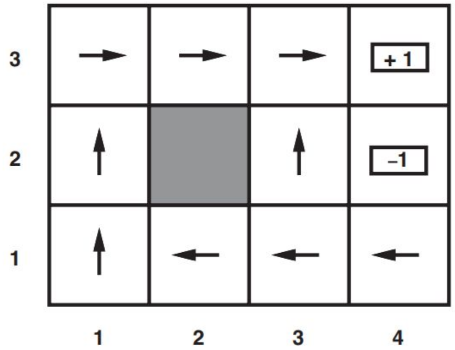
A star means an arbitrary action.

Bellman Equation

Establish a relation between U(s) and all neighboring U(s’) under the optimal policy.

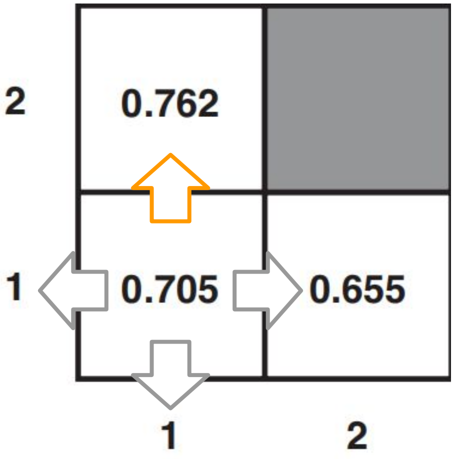
$$U(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' \mid s, a) U(s')$$

Here, s’ is a neighbor state of s.



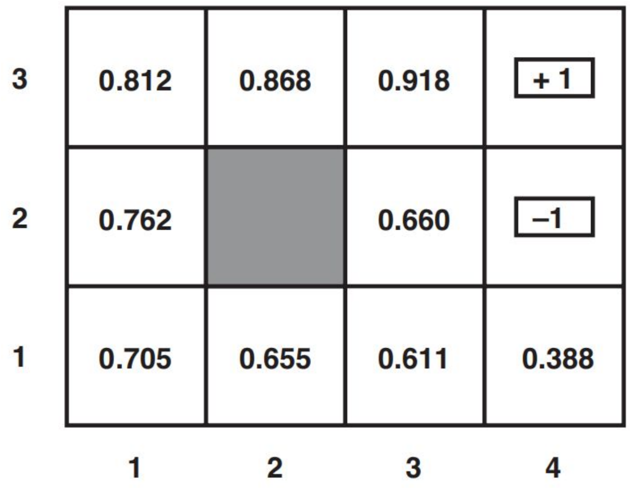
Example

$$
\begin{aligned}
U(1,1) = -0.04 + \gamma \max[\ & 0.8U(1,2) + 0.1U(2,1) + 0.1U(1,1), && 0.7456 && \boxed{(Up)} \\
& 0.9U(1,1) + 0.1U(1,2), && 0.7107 && (Left) \\
& 0.9U(1,1) + 0.1U(2,1), && 0.7 && (Down) \\
& 0.8U(2,1) + 0.1U(1,2) + 0.1U(1,1)\ ] && 0.6707 && (Right)
\end{aligned}
$$



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

Bellman equation applies to every state.



$$\text{U(1,1)} \; = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' \mid s, a) U(s')$$

$$\text{U(1,2)} \; = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' \mid s, a) U(s')$$

$$\vdots$$

$$\text{U(4,1)} \; = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' \mid s, a) U(s')$$

Markov Decision Process

Problem Statement

Given reward and transition probability, estimate the utility of each state (and optimal policy).

There are three popular techniques.

Value Iteration

Policy Iteration

Monte Carlo

Finding Utilities = Solving Bellman Equations

3	0.812	0.868	0.918	+1
2	0.762		0.660	–1
1	0.705	0.655	0.611	0.388
	1	2	3	4

$$\text{U(1,1)} = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' \mid s, a) U(s')$$

$$\text{U(1,2)} = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' \mid s, a) U(s')$$

$$\vdots$$

Not easy to solve.

$$\text{U(4,1)} = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' \mid s, a) U(s')$$

1. The Value Iteration Algorithm

1. initialize all U(s) randomly.

2. update every U(s) using the Bellman equation *concurrently*

$$U_{i+1}(s) \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' \mid s, a) U_i(s')$$

3. repeat step 2 until converge

Example

1. randomly initialize $U(1,1) = 0.3$, $U(1,2) = 0.1$, …, $U(4,1) = 0.7$.

2. update $U(1,1) = -0.04 + 0.8 * \max\{0.3, 0.6, 0.1, 0.9\} = 0.7$

$U(1,2) = -0.04 + 0.8 * \max\{0.7, 0.3, 0.5, 0.7\} = 0.5$

...

$U(4,1) = -0.04 + 0.8 * \max\{0.2, 0.8, 0.4, 0.1\} = 0.6$

3. repeat 2 until convergence, e.g., output $U(1,1)=0.6$, $U(1,2)=0.7$, …, $U(4,1)=0.1$

All values in the max function are computed based on current utilities.

Demo

This example is from the lecture slides of “Markov Decision Process and Exact Solution Methods” by Pieter Abbeel.

Demo


0.00
0.00
0.72
1.00
0.00
0.00
-1.00
0.00
0.00
0.00
0.00
VALUES AFTER 2 ITERATIONS

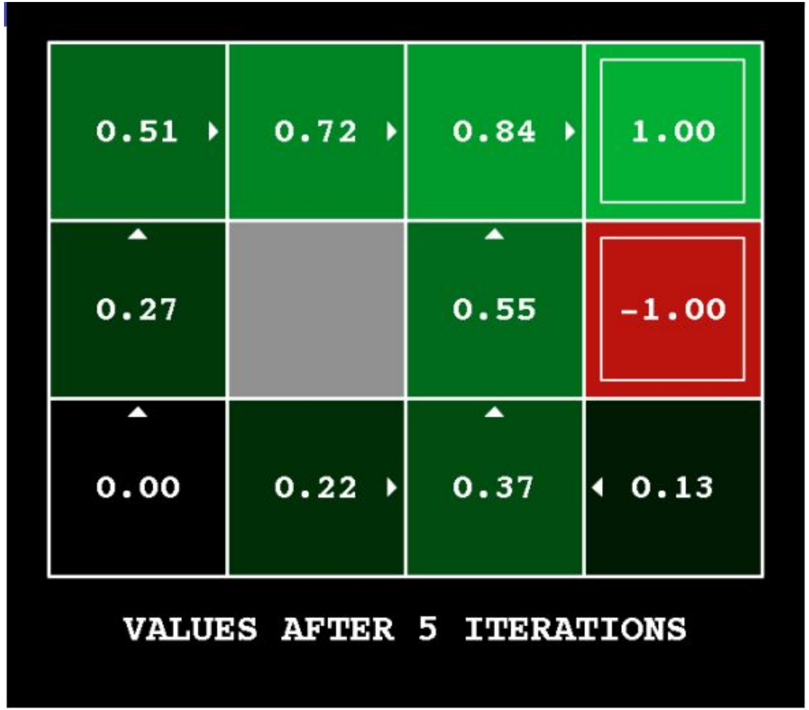
Demo

0.00
0.52
0.78
1.00
0.00
0.43
-1.00
0.00
0.00
0.00
0.00
VALUES AFTER 3 ITERATIONS

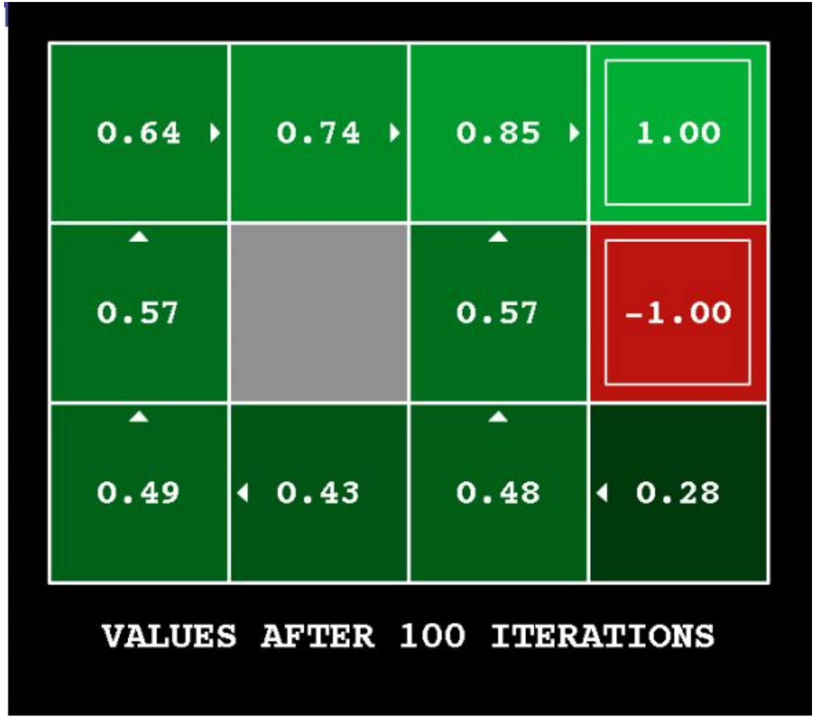
Demo


0.37
0.66
0.83
1.00
0.00
0.51
-1.00
0.00
0.00
0.31
0.00
VALUES AFTER 4 ITERATIONS

Demo


0.51
0.72
0.84
1.00
0.27
0.55
-1.00
0.00
0.22
0.37
0.13
VALUES AFTER 5 ITERATIONS

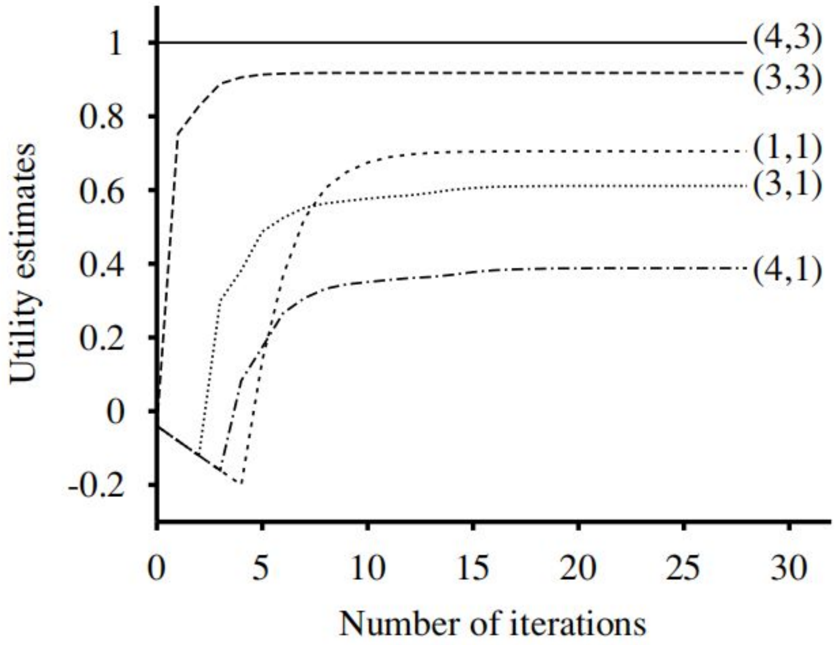
Demo


0.64
0.74
0.85
1.00
0.57
0.57
-1.00
0.49
0.43
0.48
0.28
VALUES AFTER 100 ITERATIONS

Demo


0.64
0.74
0.85
1.00
0.57
0.57
-1.00
0.49
0.43
0.48
0.28
VALUES AFTER 1000 ITERATIONS

Convergence of the Value Iteration Algorithm

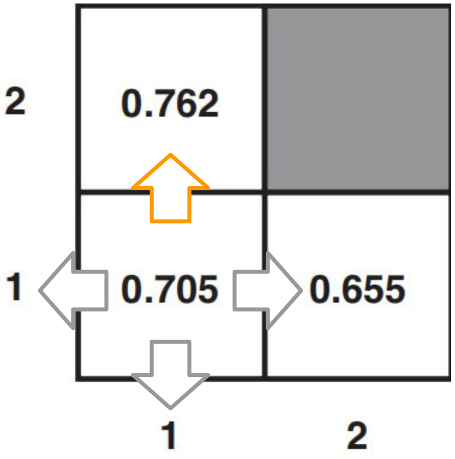


$$\max_{s \in \mathcal{S}} |V^k(s) - V^\star(s)| \leq \frac{\gamma^k R_{\max}}{1 - \gamma}$$

Convergence is guaranteed.

After convergence, identify optimal policy

$$
\begin{aligned}
U(1,1) = -0.04 + \gamma \max[\ & 0.8U(1,2) + 0.1U(2,1) + 0.1U(1,1), && 0.7456 && \boxed{(Up)} \\
& 0.9U(1,1) + 0.1U(1,2), && 0.7107 && (Left) \\
& 0.9U(1,1) + 0.1U(2,1), && 0.7 && (Down) \\
& 0.8U(2,1) + 0.1U(1,2) + 0.1U(1,1)\ ] && 0.6707 && (Right)
\end{aligned}
$$



The action that gives the largest expected utility is the optimal action (policy).

2. The Policy Iteration Algorithm

Main idea is to optimize policy and utility alternately. Note that with a fixed policy, Bellman equations become linear and are hence very easy to solve.

1. Randomly initialize policy.

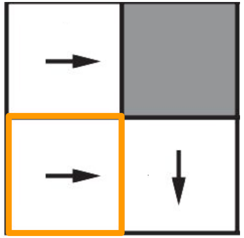
2. Fix policy, optimize utility by solving (linear) Bellman equations.

3. Fix utility, optimize policy based on the (original) Bellman equation.

4. Repeat 2-4 until convergence.

Step 2. Fix policy and optimize utility.

$$
\begin{aligned}
U(1,1) = -0.04 + \gamma \max[\; & 0.8U(1,2) + 0.1U(2,1) + 0.1U(1,1), && 0.75 \quad (Up)\\
& 0.9U(1,1) + 0.1U(1,2), && 0.71 \quad (Left)\\
& 0.9U(1,1) + 0.1U(2,1), && 0.70 \quad (Down)\\
& \boxed{0.8U(2,1) + 0.1U(1,2) + 0.1U(1,1)\;]. \quad 0.67 \quad (Right)}
\end{aligned}
$$



$$
U(1,1) = -0.04 + \gamma \; [\; 0.8U(2,1) + 0.1U(1,2) + 0.1U(1,1)\;]. \qquad (Right)
$$

Step 2. Fix policy and optimize utility.

Now we solve n linear equations for the utilities of n states. (easy)

$$
\begin{aligned}
U_i(1,1) &= -0.04 + 0.8U_i(1,2) + 0.1U_i(1,1) + 0.1U_i(2,1) \\
U_i(1,2) &= -0.04 + 0.8U_i(1,3) + 0.2U_i(1,2)\ , \\
&\vdots
\end{aligned}
$$

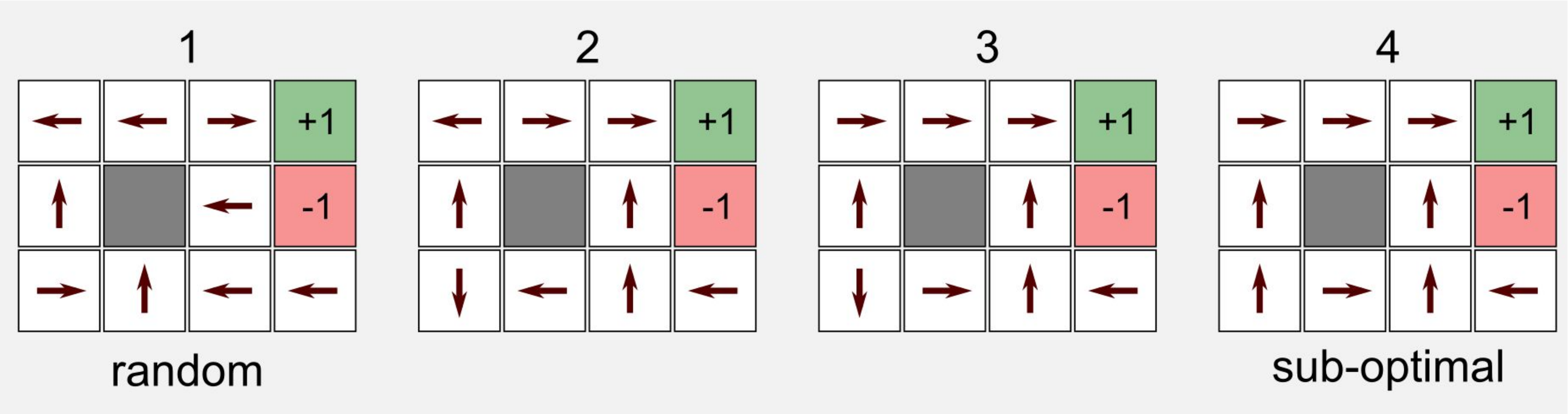
Step 3. Fix utility and optimize policy.

Given utility, we can identify optimal policy using the Bellman equation.

$$
\begin{aligned}
U(1,1) = -0.04 + \gamma \max[\ & 0.8U(1,2) + 0.1U(2,1) + 0.1U(1,1), && 0.75 && \boxed{(Up)} \\
& 0.9U(1,1) + 0.1U(1,2), && 0.71 && (Left) \\
& 0.9U(1,1) + 0.1U(2,1), && 0.70 && (Down) \\
& 0.8U(2,1) + 0.1U(1,2) + 0.1U(1,1)\ ]. && 0.67 && (Right)
\end{aligned}
$$

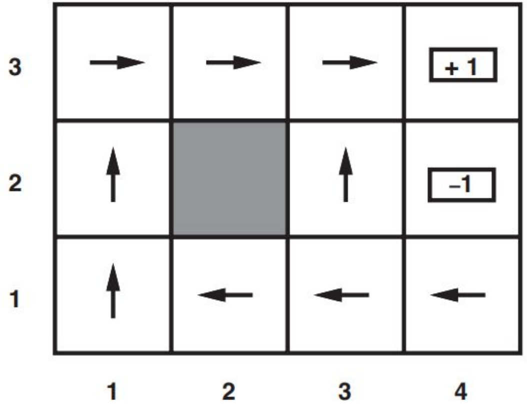
The action that gives the largest expected utility is the optimal action (policy).

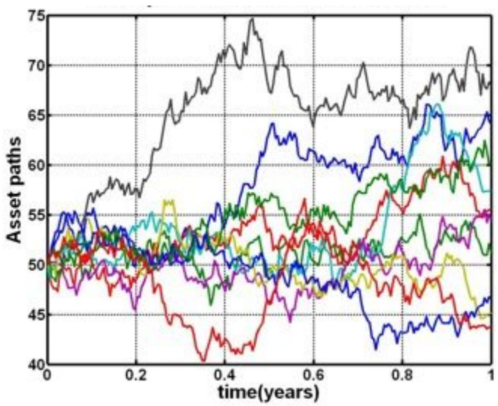
Demo

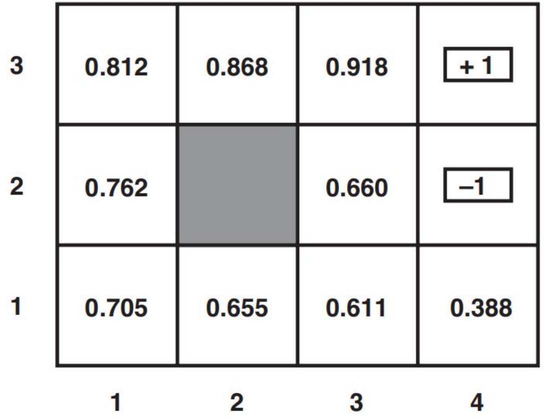


3. The Monte Carlo Algorithm

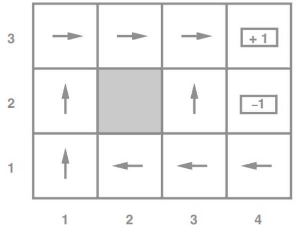
Idea: simply run experiments to estimate utilities. (not designed to find policy)



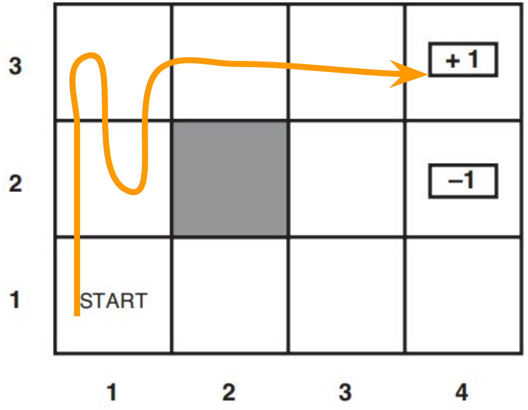


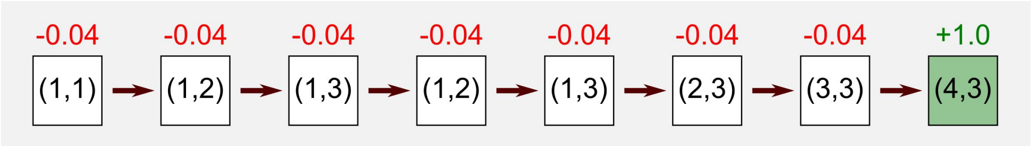


Example: Estimate U(1,1)



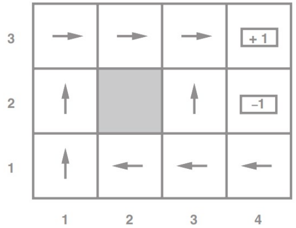
1st trial: U(1,1) = 7*(-0.04) + 1 = 0.72



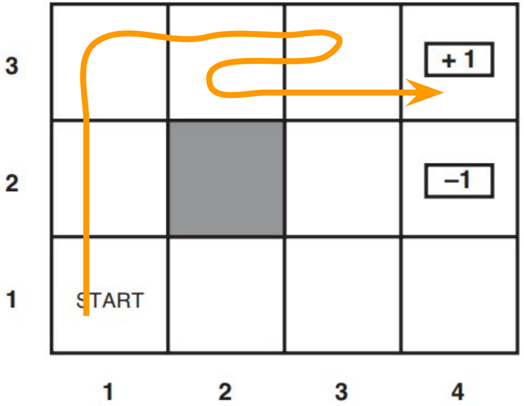


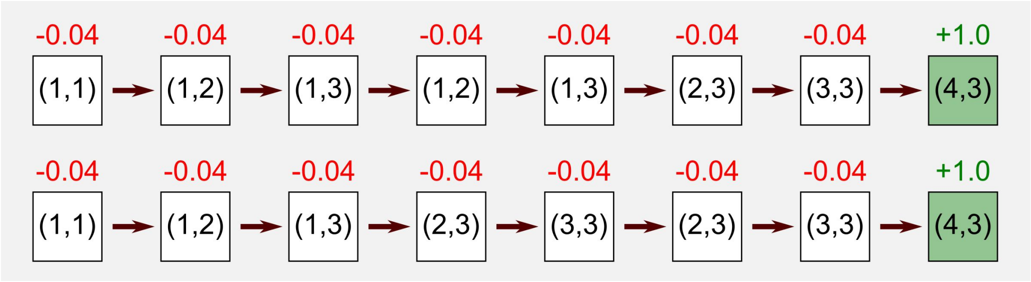
For simplicity, assume γ=1 in this example.

Example: Estimate U(1,1)

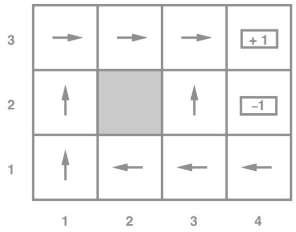


2nd trial: $U(1,1) = 7*(-0.04) + 1 = 0.72$

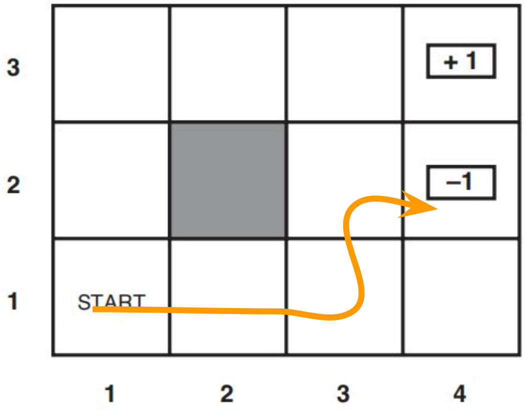


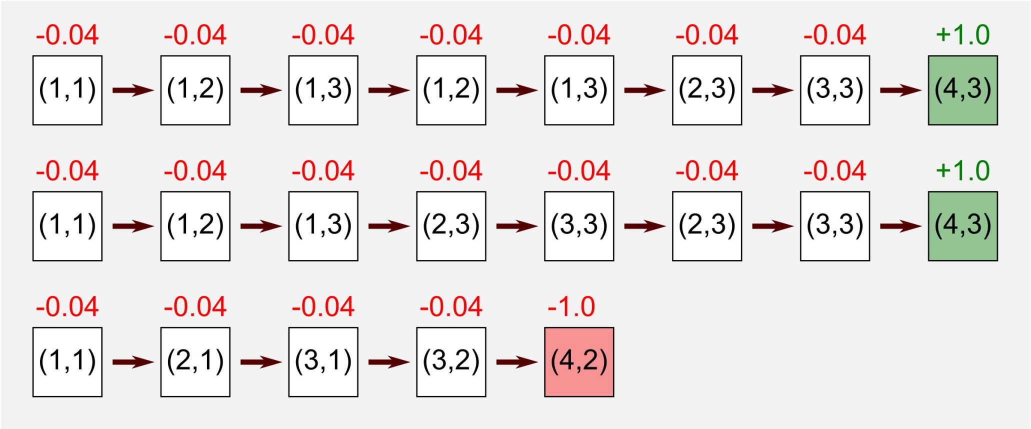


Example: Estimate U(1,1)

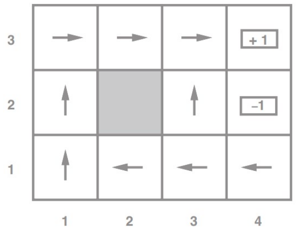


3rd trial: U(1,1) = 4*(-0.04) - 1 = -1.16

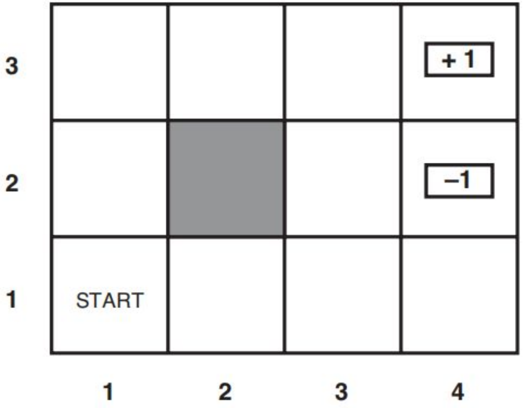


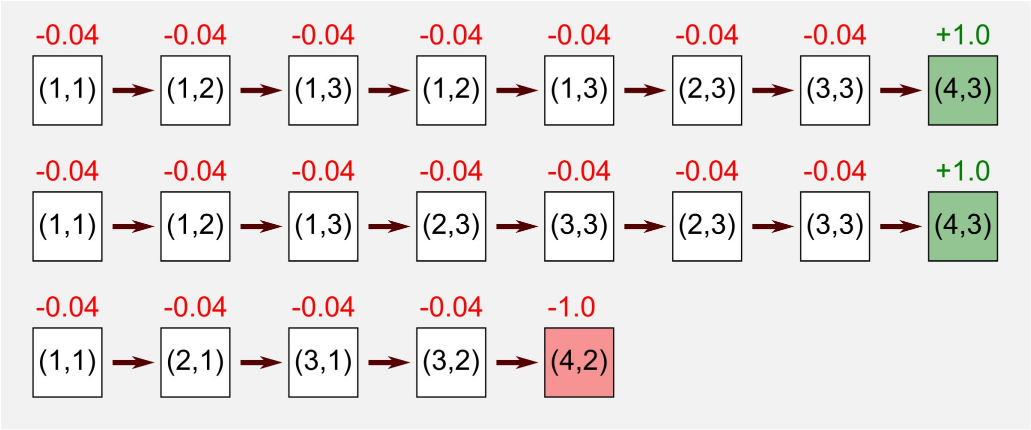


Example: Estimate U(1,1)



Average $U(1,1) \approx (0.72+0.72-1.16) / 3 = 0.093$

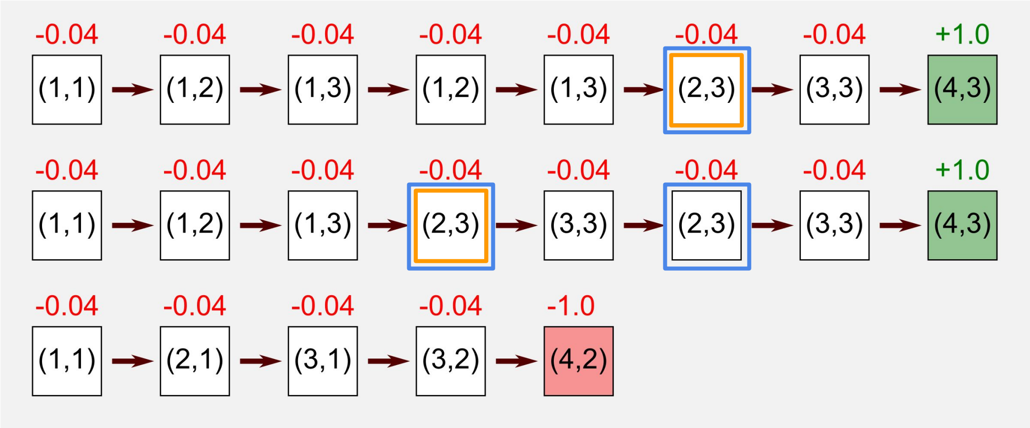




First-visit MC vs Every-visit MC

First-visit MC: estimate U(s) based on trials where s is the starting point.

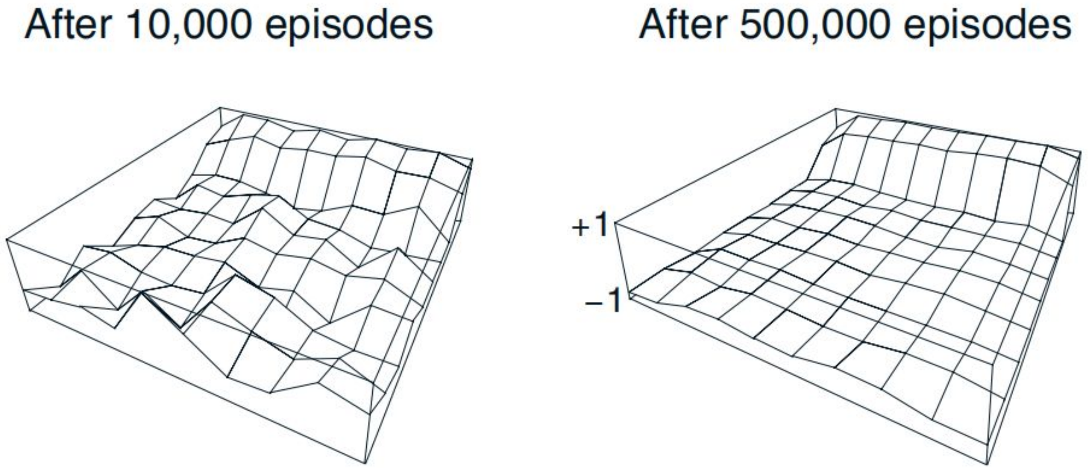
Every-visit MC: estimate U(s) based on every trial that includes s.



In every-visit MC, we can use these results to estimate U(2,3) as well.

Convergence

MC converges to true utility as the number of visits approaches infinity.



Apply MC + Policy Iteration to Optimize Policy

1. Randomly initialize {policy}.

2. Fix policy, estimate utility using MC. (instead of solving Bellman equations)

3. Fix utility, optimize policy using the Bellman equation.

4. Repeat 2-4 until convergence.