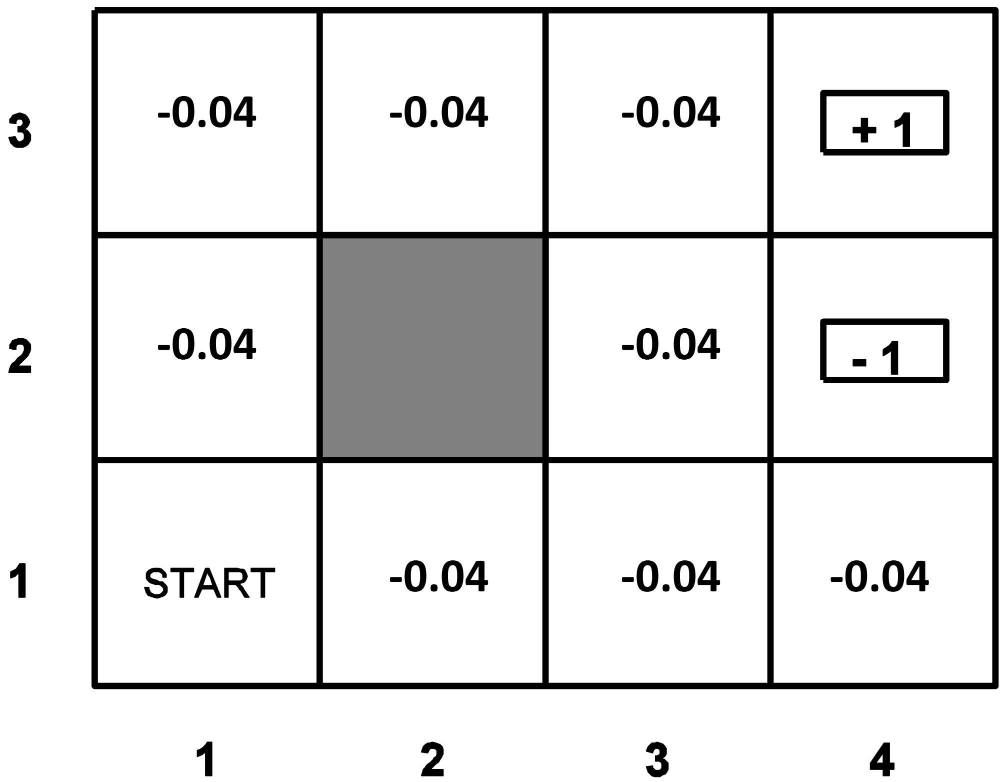
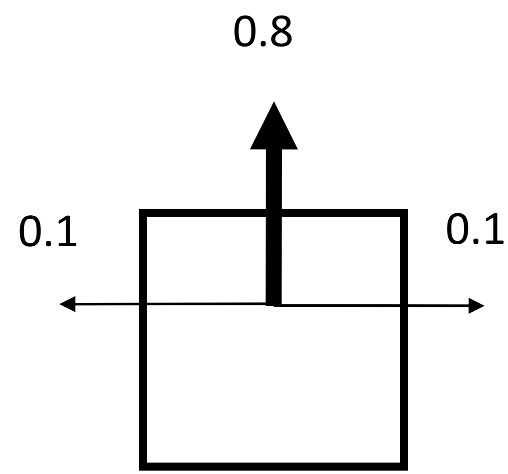
Policy Search

3	-0.04	-0.04	-0.04	+ 1
2	-0.04		-0.04	- 1
1	START	-0.04	-0.04	-0.04
	1	2	3	4





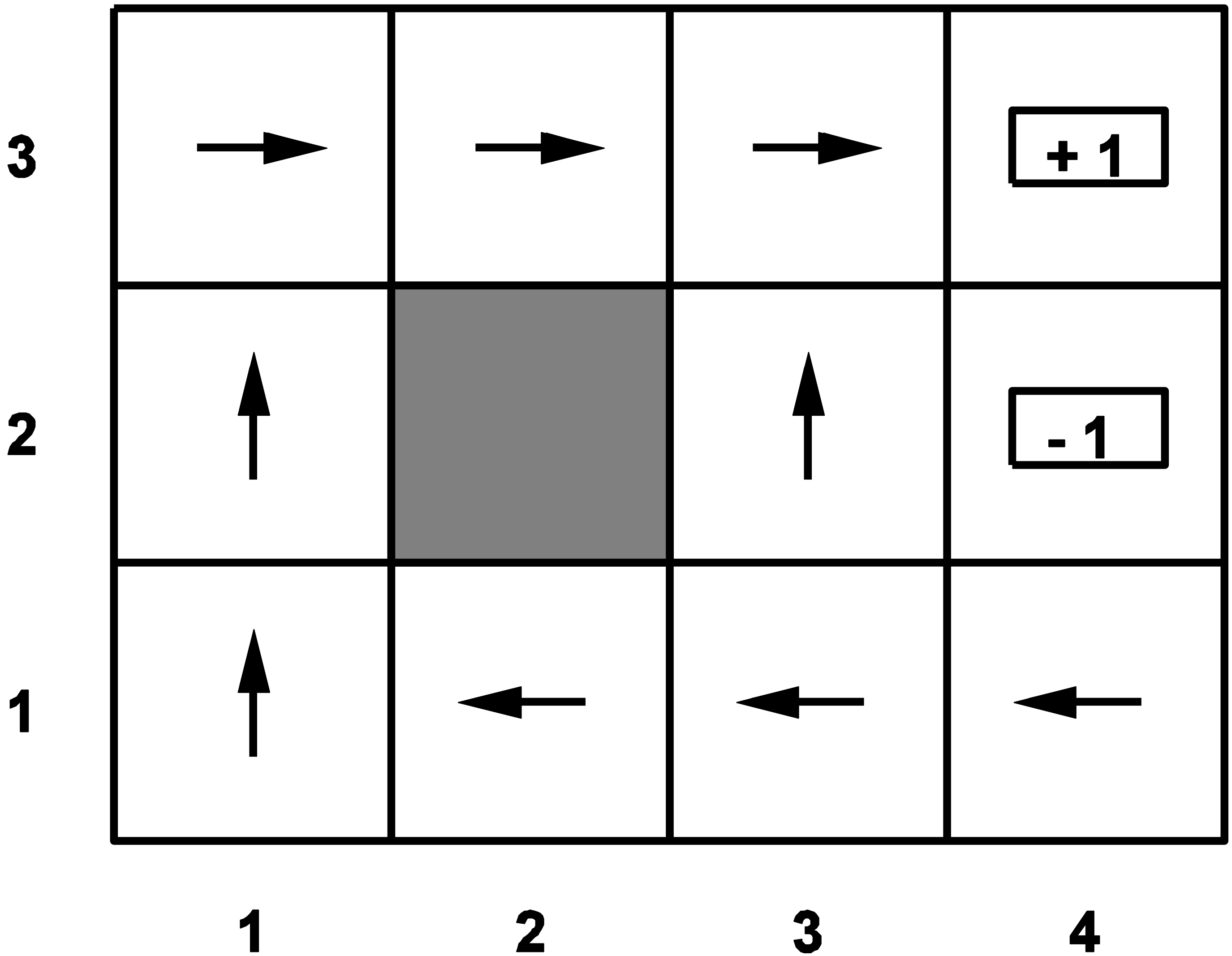
- move into desired direction with prob 80%
- move 90 degrees to left with prob 10%
- move 90 degrees to right with prob 10%

What is the probability that [up, up, right, right, right] ends in (4,3)?

A=1 B=0.32768 C=0.32776 D=0.5

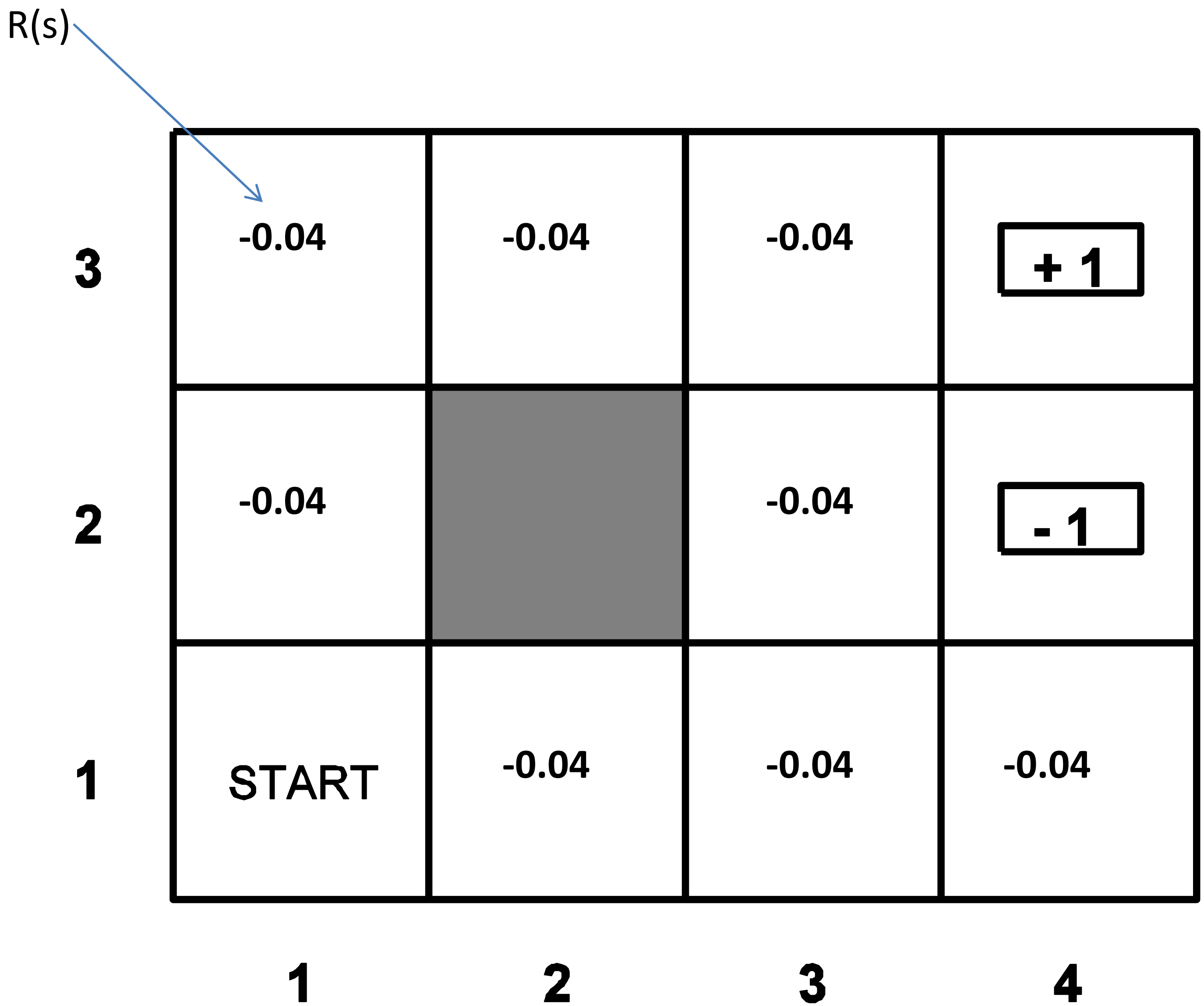
We still assume the problem is fully observable – the agent knows where it is and what the rewards are…

A Policy



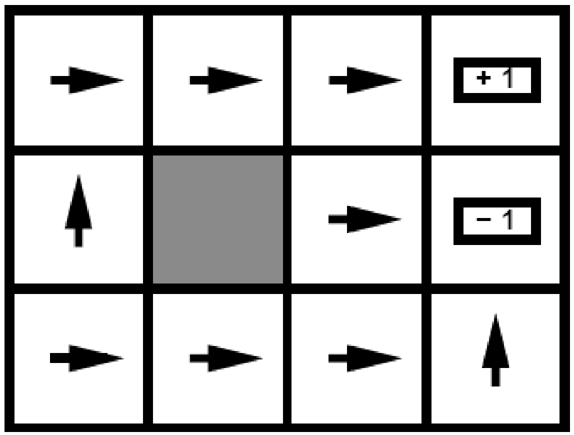
Other representations?

- How can a policy be represented?
  - An arrow for every grid cell
  - What about a continuous world?
    - Θ(x,y)
    - NN
    - C++ function

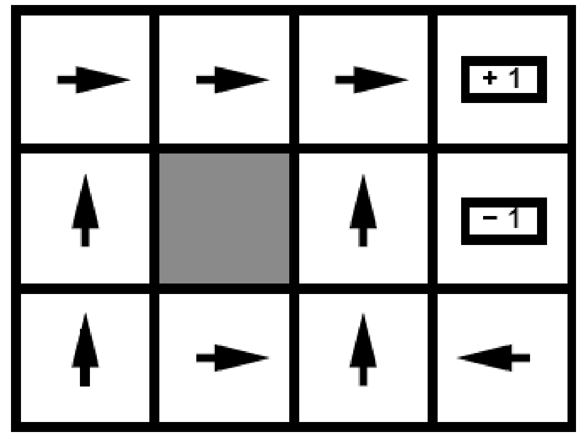


- R(s) is the short term reward in state s
- $U^{\pi}(s)$ is the long term reward when following policy π from state s

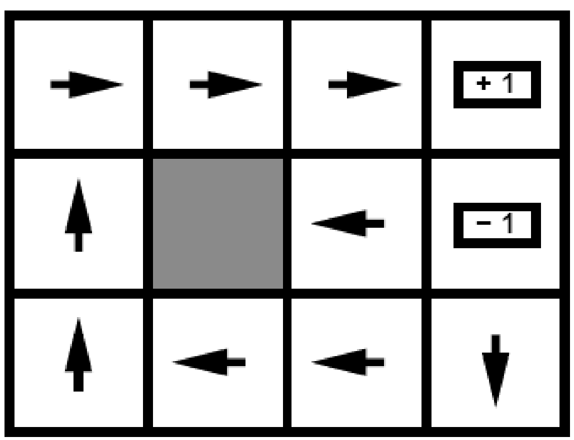
Optimal Policies for Other Rewards



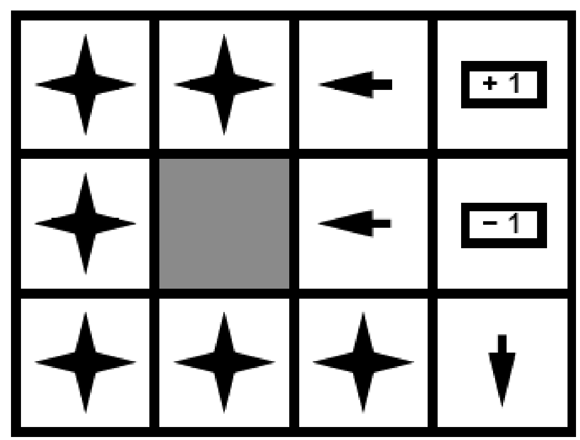
$R(s) < -1.6284$



$-0.4278 < R(s) < -0.0850$



$-0.0221 < R(s) < 0$



$R(s) > 0$

Markovian Model

- Model:
  - Initial state: S_0
  - Transition function: $T(s,a,s')$
    → $T(s,a,s')$ is the probability of moving from state s to s' when executing action a.
  - Reward function: $R(s)$
    → Real valued reward that the agent receives for entering state s.
- Assumptions
  - Markov property: $T(s,a,s')$ and $R(s)$ only depend on current state s, but not on any states visited earlier.
  - Extension: Function R may be non-deterministic as well

Markov Decision Process

- Representation of Environment:
  - finite set of states S
  - set of actions A for each state s in S
- Process
  - At each discrete time step, the agent
    - observes state s_t in S and then
    - chooses action a_t in A.
  - After that, the environment
    - gives agent an immediate reward r_t
    - changes state to s_{t+1} (can be probabilistic)

Utilities

- Rating a state sequence $[s_0, s_1, s_2, \ldots]$
  - $U([s_0,\ldots,s_N]) = \Sigma_i R(s_i)$
  - Additive rewards:
    - $U_h([s_0, s_1, s_2, \ldots]) = R(s_0) + R(s_1) + R(s_2) + \ldots$
- We want preferences to be **stationary**
  - If $[s_0, s_1, s_2, \ldots]$ better than $[s_0, s'_1, s'_2, \ldots]$ implies $[s_1, s_2, \ldots]$ better than $[s'_1, s'_2, \ldots]$
- Reward vs. Utility

Discounted Utility

- Problem:
  - What happens to utility value when
    - either the state space has no terminal states
    - or the policy never directs the agent to a terminal state

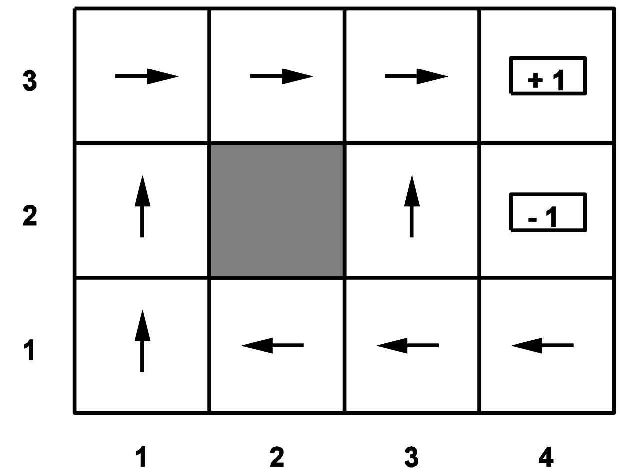
  → Utility becomes infinite
- Solution
  - Use discounted utility
    → closer rewards count more than awards far in the future

  → finite utility even for infinite state sequences

  $U([s_0,\ldots,s_N]) = \Sigma_i\, \gamma^i\, R(s_i) \leq \Sigma_i\, \gamma^i\, R_{max} = R_{max} / (1 - \gamma)$

Policy

- Definition:
  - A policy π describes which action an agent should select in each state
  - $a=\pi(s)$



- Utility of a policy
  - For now additive utility
  - Let $P([s_0,\dots,s_N] \mid \pi, s_0)$ be the probability of state sequence $[s_0,\dots,s_N]$ when following policy π from state s_0
  - Expected utility: $U^\pi(s) = \Sigma\, U([s_0,\dots,s_N])\, P([s_0,\dots,s_N] \mid \pi, s_0)$ → measure of quality of policy π
  - Optimal policy π*: Policy with maximal $U^\pi(s)$ in each state s

Utility ⇔ Policy

- Equivalence:
  - If we know the utility U(s) of each state, we can derive the correseponding policy:

    $$\pi^*(s) = \text{argmax}_a \; \Sigma_{s'} \, T(s, a, s') \, U(s')$$

  - If we know the policy π, we can compute the corresponsing utility of each state:

    PolicyEvaluation algorithm

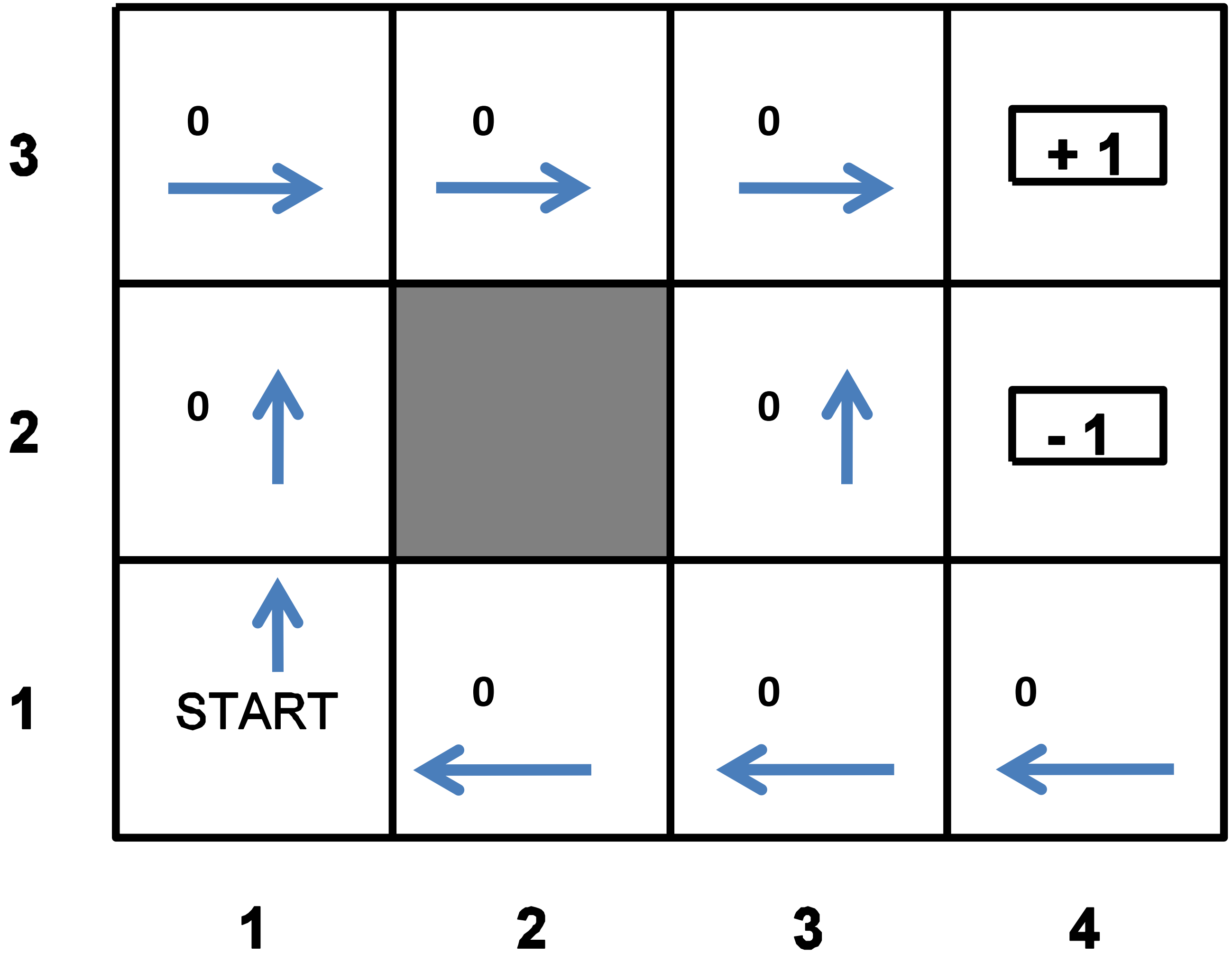
  Bellman Equation:

  $$U(s) = R(s) + \gamma \, \max_a \Sigma_{s'} \, T(s, a, s') \, U(s')$$

  Optimal Utility ⇔ Optimal Policy

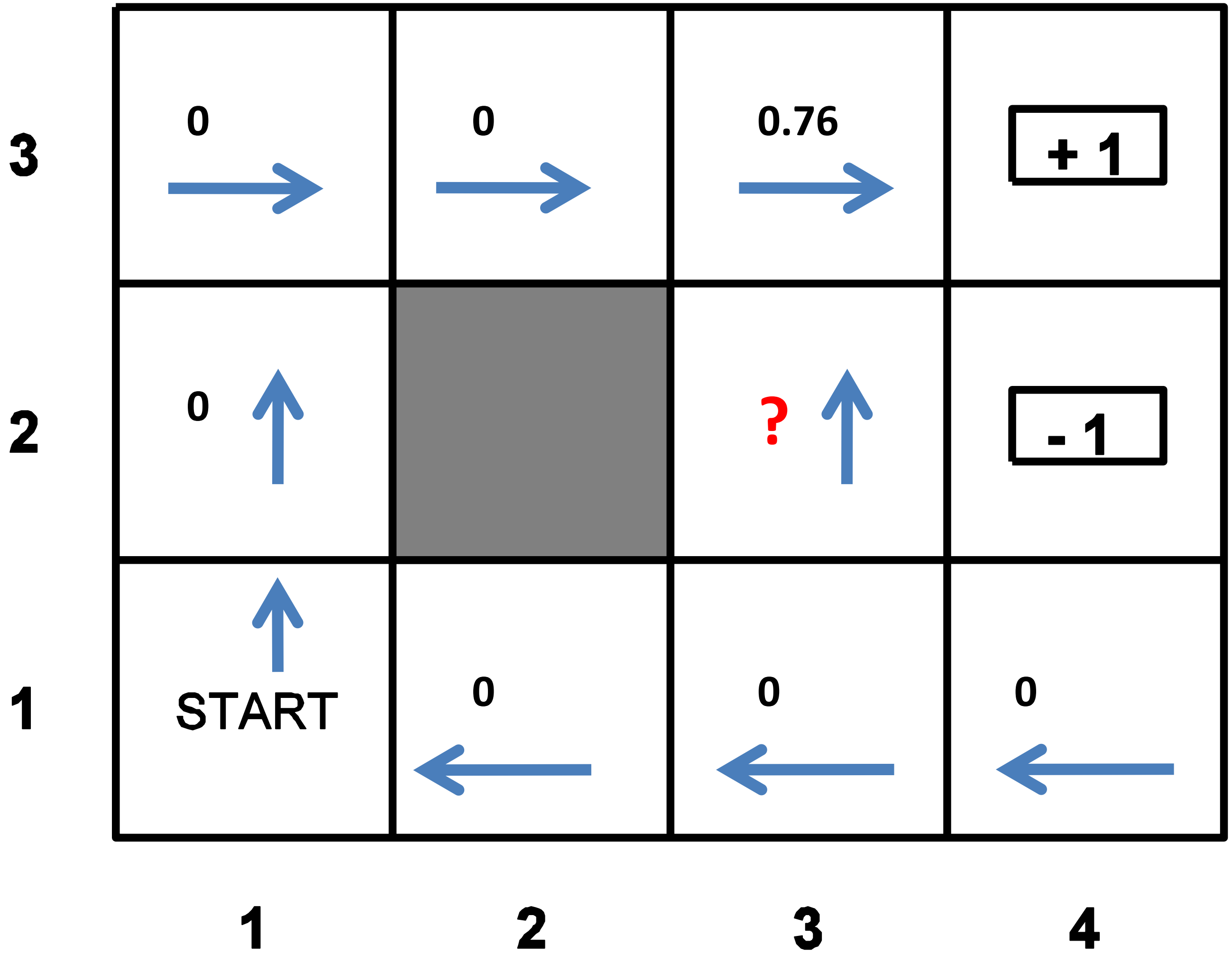
How to Compute the Utility for a given Policy?

- Definition:
  - $U^\pi(s) = \Sigma \, [\overset{\text{Utility of path}}{\boxed{\Sigma_i \, \gamma^i \, R(s_i)}}] \, P(\overset{\text{Likelihood of path}}{\boxed{[s_0, s_1, \ldots]}} \mid \pi, s_0 = s)$
- Recursive computation:
  - $U^\pi(s) = R(s) + \gamma \, \Sigma_{s'} \, T(s, \pi(s), s') \, U^\pi(s')$
- What is $P([s_0, s_1, \ldots] \mid \pi)$?
  - $P([s_0, s_1, \ldots] \mid \pi) = T(s_0, \pi(s_0), s_1) * T(s_1, \pi(s_1), s_2) * \ldots$

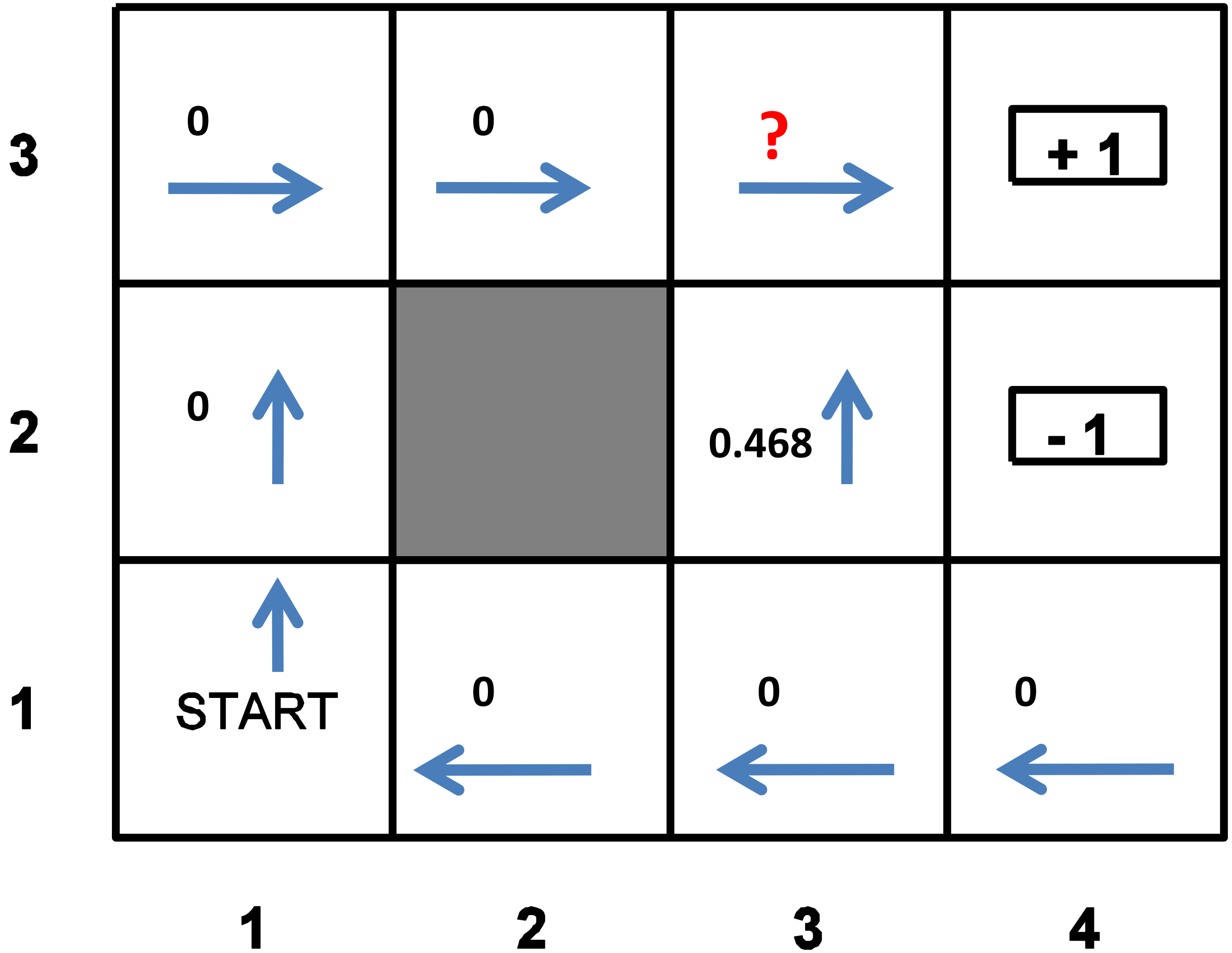


• $U^{\pi}((3,3)) \leftarrow -0.04 + 0.8 * 1 + 0.1 * 0 + 0.1 * 0 = 0.76$

R(s)= -0.04 everywhere except goals



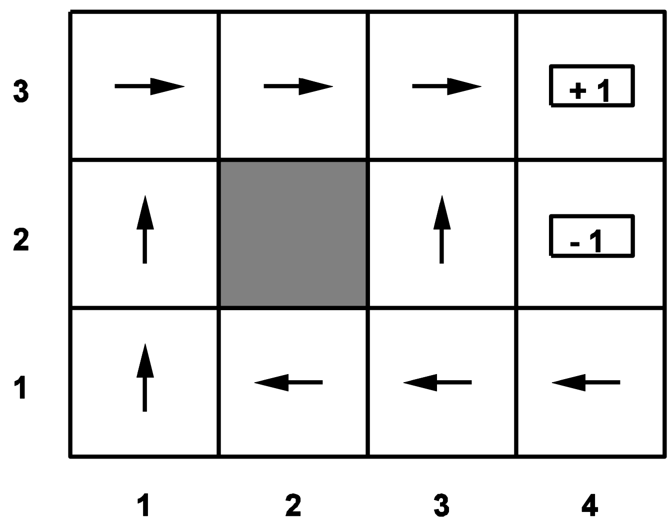
•$U^{\pi}((3,2))$ = ? A= -0.04 B=0.312 C=0.428 D=0.468

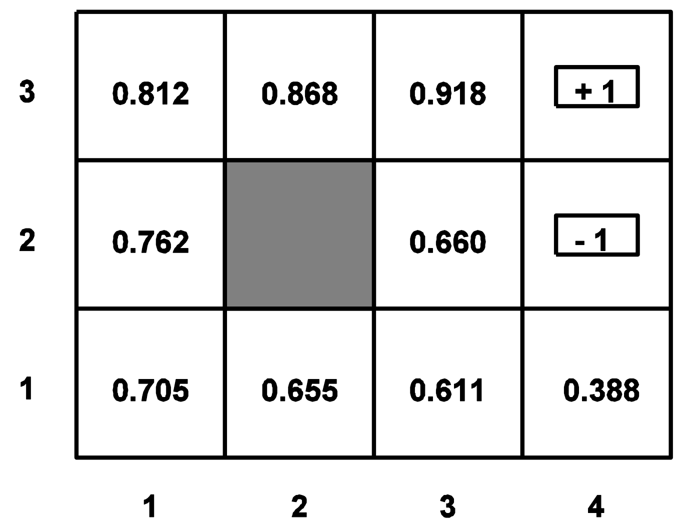


•$U^{\pi}((3,3))$ = ?   A= -0.04   B=0.760   C=0.883   D=0.885

Convergence

$$U^{\pi}(s) = R(s) + \gamma\, \Sigma_{s'}\, T(s, \pi(s), s')\, U^{\pi}(s')$$





Here: γ=1.0, R(s)=-0.04

Bellman Update

- Goal: Solve set of n=|S| equations (one for each state)

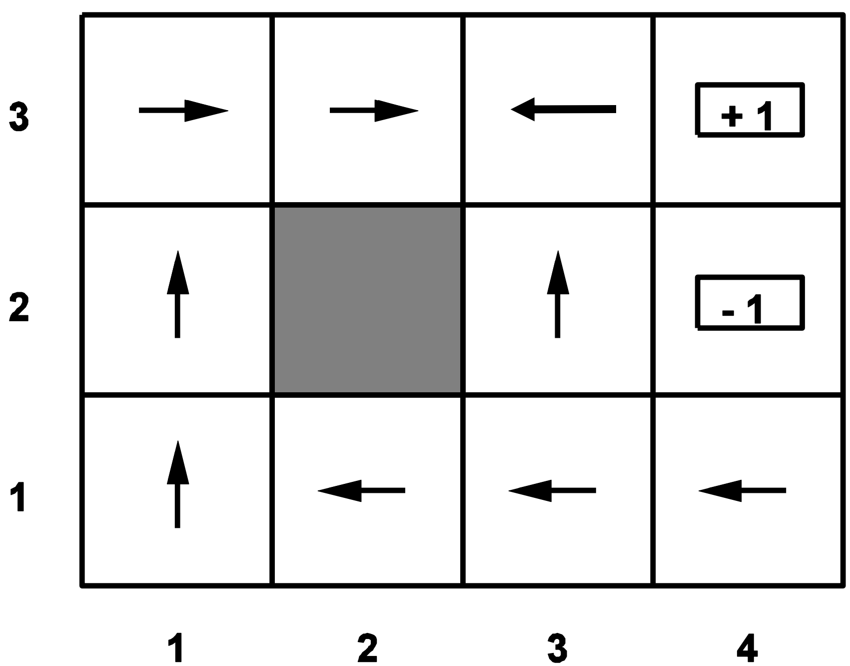
$$U^{\pi}(s_0) = R(s_0) + \gamma \, \Sigma_{s'} \, T(s_0, \pi(s), s') \, U^{\pi}(s')$$
$$\ldots$$
$$U^{\pi}(s_n) = R(s_n) + \gamma \, \Sigma_{s'} \, T(s_n, \pi(s), s') \, U^{\pi}(s')$$

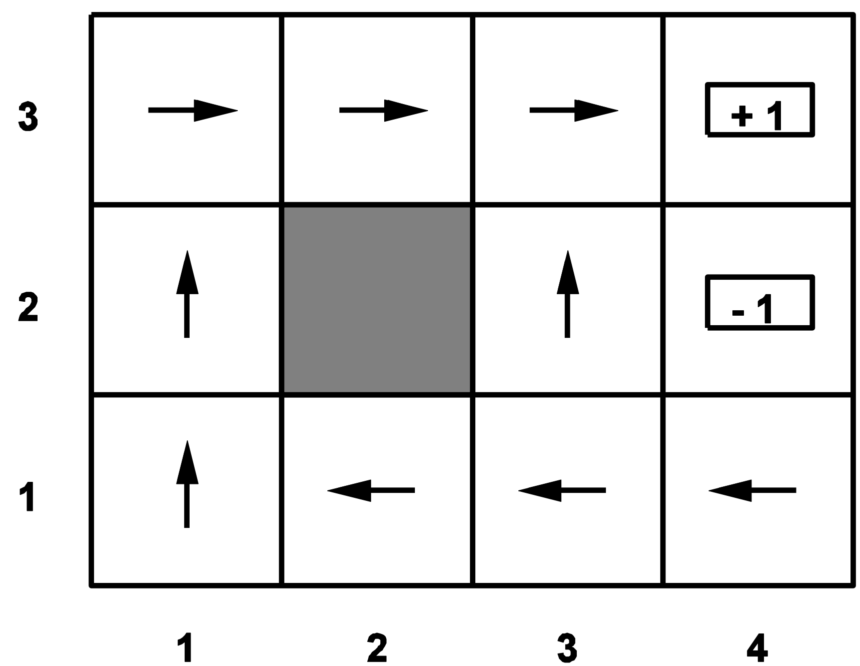
- Is this a set of linear equations? Why or why not?

- Algorithm [Policy Evaluation] (fix π):
  - i=0; $U^{\pi}_0(s)=0$ for all s
  - repeat
    - i = i +1
    - for each state s in S do
      - $U^{\pi}_i(s) = R(s) + \gamma \, \Sigma_{s'} \, T(s, \pi(s), s') \, U^{\pi}_{i-1}(s')$
    - endfor
  - until difference between U^{π}_i and U^{π}_{i-1} small enough
  - return U^{π}_i

How to Find the Optimal Policy π^*?

- Is policy π optimal? How can we tell?
  - If π is not optimal, then there exists some state where
    $$\pi(s) \neq \text{argmax}_a \sum_{s'} T(s, a, s')\, U^{\pi}(s')$$
  - How to find the optimal policy π^*?





How to Find the Optimal Policy π^*?

- Algorithm [Policy Iteration]:
  - repeat
    - U^π = PolicyEvaluation(π,S,T,R)
    - for each state s in S do
      - If [$\max_a \boldsymbol{\Sigma_{s'}}$ **T(s, a, s') U^π(s') > $\Sigma_{s'}$ T(s,** π(s)**, s') U^π(s')] then**
        - » π(s) **=** argmax_a **$\Sigma_{s'}$ T(s, a, s') U^π(s')**
    - **endfor**
  - until π does not change any more
  - return π

Value Iteration

- Optimal Utility ⇔ Optimal Policy
- Find optimal utility directly

1	-2	5
7	**?**	6
3	6	5

↑ **A**

← **B**

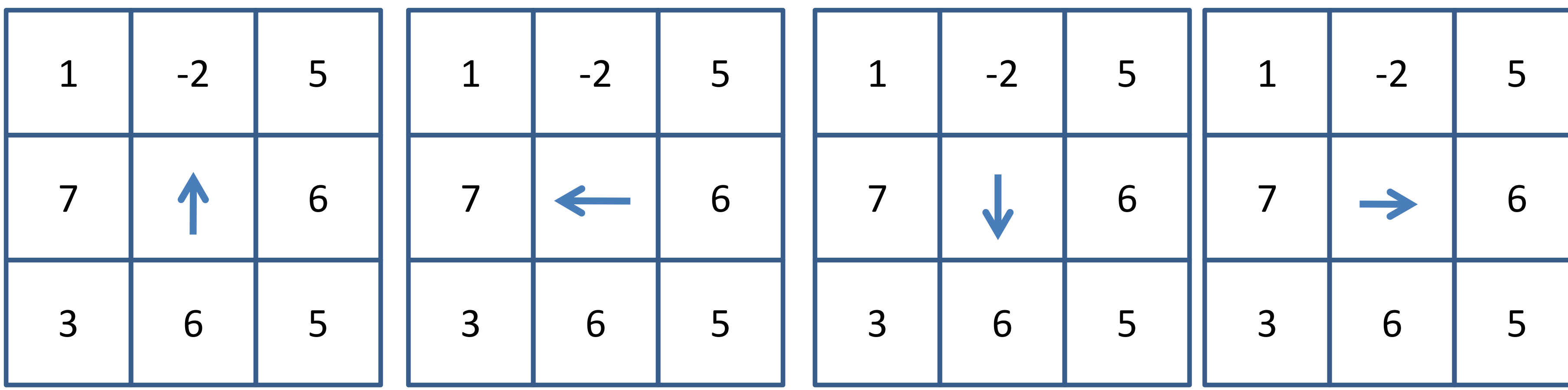
↓ **C**

→ **D**

What is the best utility possible for the center?

Value Iteration

- Optimal Utility ⇔ Optimal Policy
- Find optimal utility directly



-0.04 + 0.8*-2 + 0.1*7 + 0.1*6
=-0.34

-0.04 + 0.8*7 + 0.1*-2 + 0.1*6
=5.96

-0.04 + 0.8*6 + 0.1*7 + 0.1*6
=6.06

-0.04 + 0.8*6 + 0.1*6 + 0.1*-2
=5.16

Value Iteration Algorithm

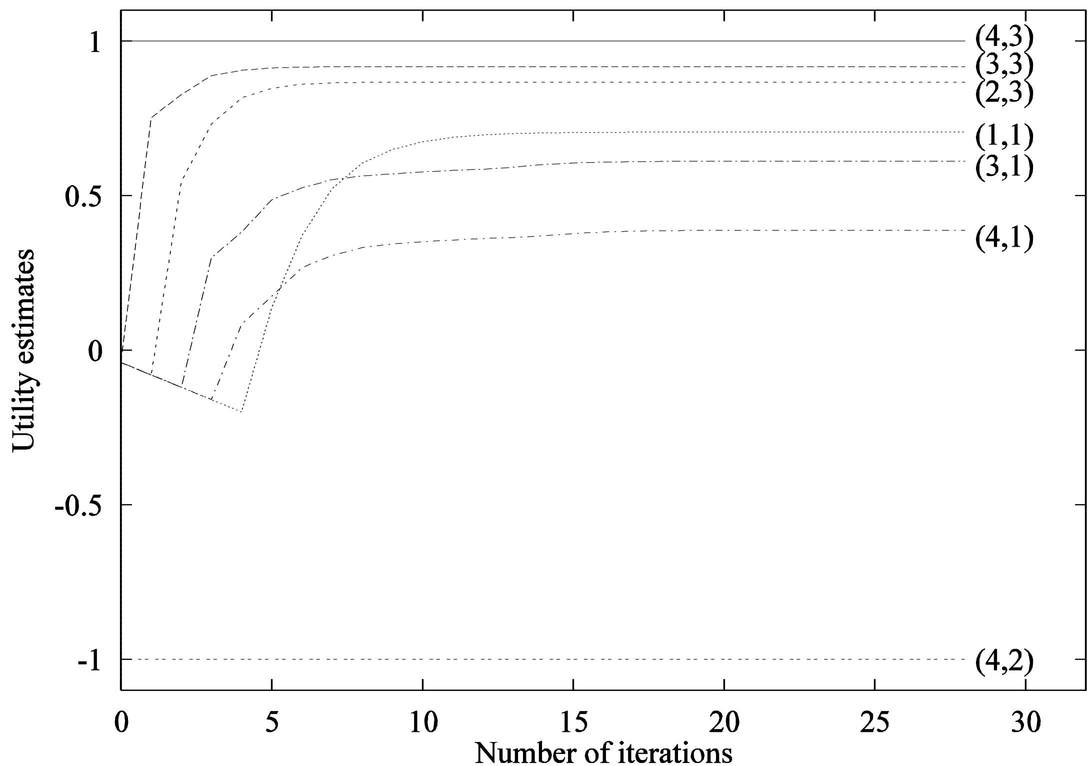
- Algorithm [Value Iteration]:
  - $i=0$; $U_0(s)=0$ for all s
  - repeat
    - $i = i + 1$
    - for each state s in S do
      - $U_i(s) = R(s) + \gamma \max_a \Sigma_{s'} T(s, a, s') U_{i-1}(s')$
    - endfor
  - until difference between U_i and U_{i-1} small enough
  - return U_i

→ derive optimal policy via $\pi^*(s) = \text{argmax}_a \Sigma_{s'} T(s, a, s') U(s')$

Two approaches to finding π*

- Policy Iteration
  - Local search: Start with random policy
  - Evaluate a candidate policy using bellman update
  - $U^\pi(s_0) = R(s_0) + \gamma \Sigma_{s'} T(s_0, \pi(s), s') U^\pi(s')$
  - Linear set of equations
  - Confirm policy is optimal or make changes
- Value Iteration
  - find optimal utilities by iteratively solving
    - $U_i(s) = R(s) + \gamma \max_a \Sigma_{s'} T(s, a, s') U_{i-1}(s')$
  - Derive optimal policy to follow optimal utilities

Convergence of Value Iteration



- Value iteration is guaranteed to converge to optimal U for $0 \leq \gamma < 1$
- Faster convergence for smaller γ
- Guaranteed to reach equilibrium (see textbook)