CS 4700: Foundations of Artificial Intelligence

Spring 2020

Prof. Haym Hirsh

Lecture 17

March 4, 2020

Markov Decision Process

Expands on State Space Search:

- States: S (including some Initial State s_0)
- Actions: A

$$P(s' \mid s, a)\ s \in S,\ a \in A$$

- Reward Function:

$$R: S \times A \times S \rightarrow \mathbb{R}$$

- Goal State/Condition: Get lots of reward over time

What Does “Get Lots of Reward” Mean?

One approach: Cumulative reward

Given: a sequence of states [s_0, s_1, …, s_t, …] using actions [a_0, a_1, …, a_t, …]

Cumulative reward:

$$\sum_{t=0}^{\infty} \mathrm{R}(s_t, a_t, s_{t+1})$$

What Does “Get Lots of Reward” Mean?

One approach: Cumulative reward

Given: a policy Π (where $a_t = \Pi(s_t)$, $s_{t+1} = \text{apply}(\Pi(s_t), s_t)$)

Actions are probabilistic

Cumulative reward:

$$E\left[\sum_{t=0}^{\infty} \text{R}(s_t, \Pi(s_t), \text{apply}(\Pi(s_t), s_t))\right]$$

What Does “Get Lots of Reward” Mean?

One approach: Cumulative reward

Given: a policy Π (where $a_t = \Pi(s_t), s_{t+1} = \text{apply}(\Pi(s_t), s_t)$)

Actions are probabilistic

Cumulative reward:

$$U_C^{\Pi}(s_0) = E\left[\sum_{t=0}^{\infty} \text{R}(s_t, \Pi(s_t), \text{apply}(\Pi(s_t), s_t))\right]$$

Markov Decision Process

Expands on State Space Search:

- States: S (including some Initial State s_0)
- Actions: A

$$P(s' \mid s, a)\ s \in S,\ a \in A$$

- Reward Function:

$$R: S \times A \times S \rightarrow \mathbb{R}$$

- Goal State/Condition: Get lots of reward over time

Markov Decision Process

Expands on State Space Search:

- States: S (including some Initial State s_0)
- Actions: A

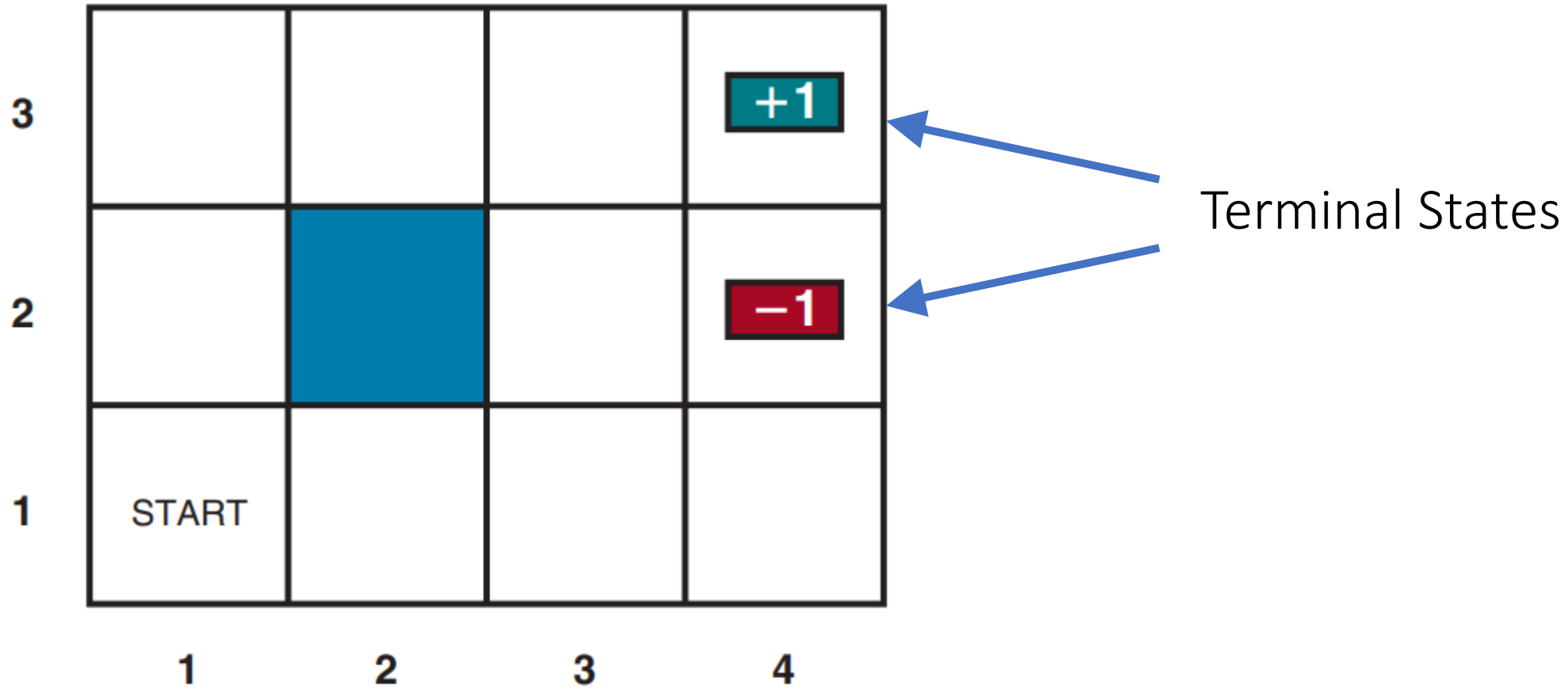
$$P(s' \mid s, a)\ s \in S,\ a \in A$$

- Reward Function:

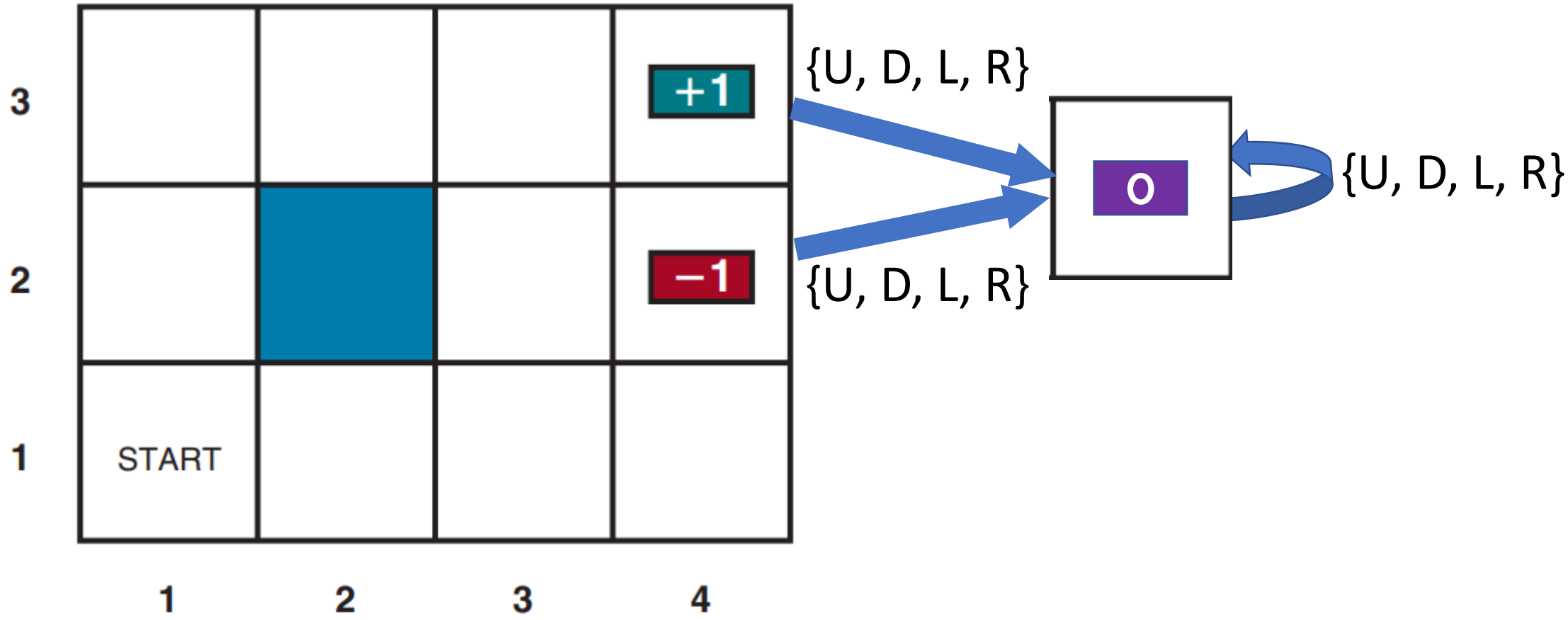
$$R: S \times A \times S \rightarrow \mathbb{R}$$

- Goal State/Condition: Find a policy Π that maximizes $U_C^{\Pi}(s_0)$

Example (Figure 17.1)

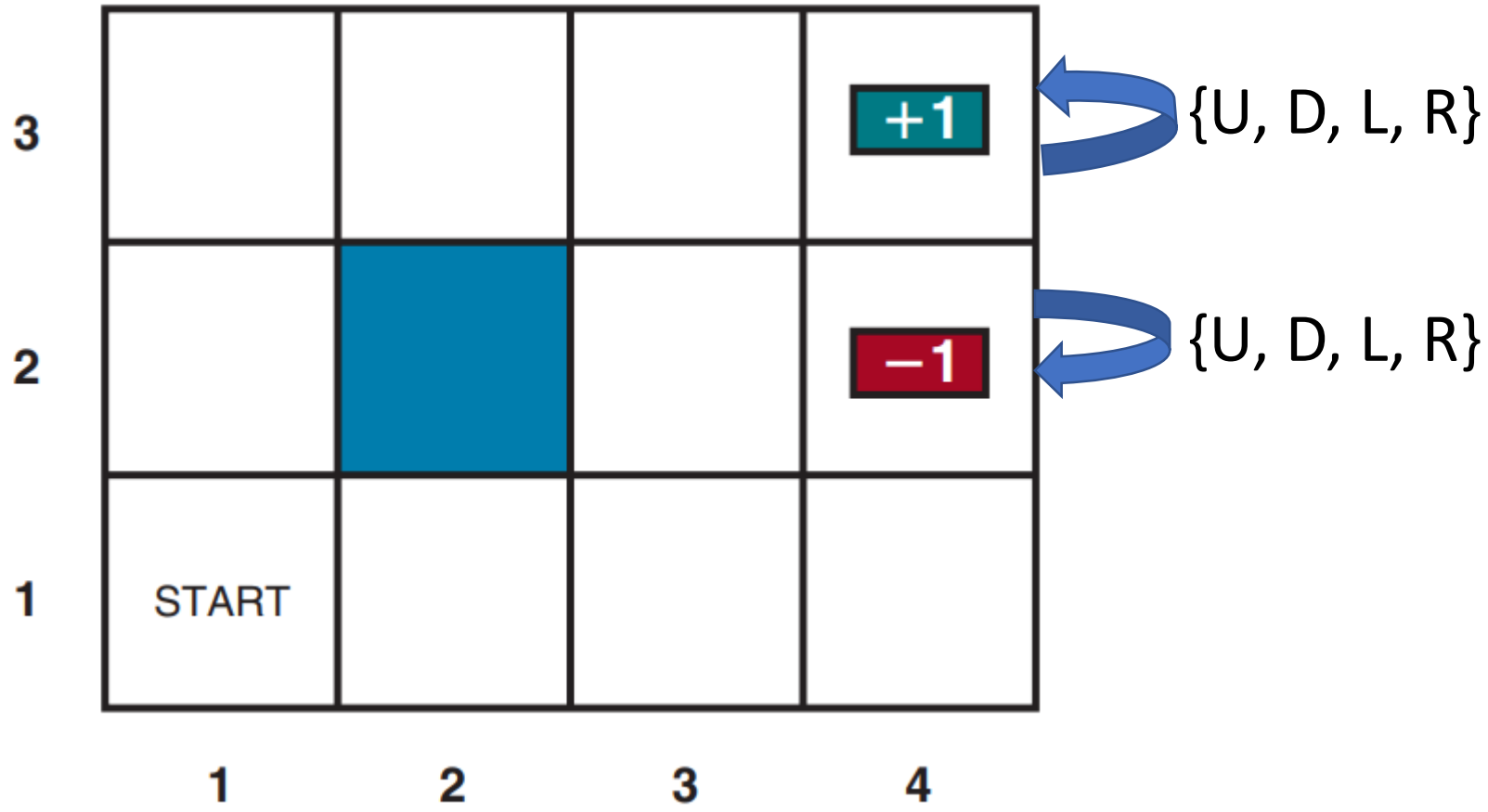


Example: Version 1 – End of problem



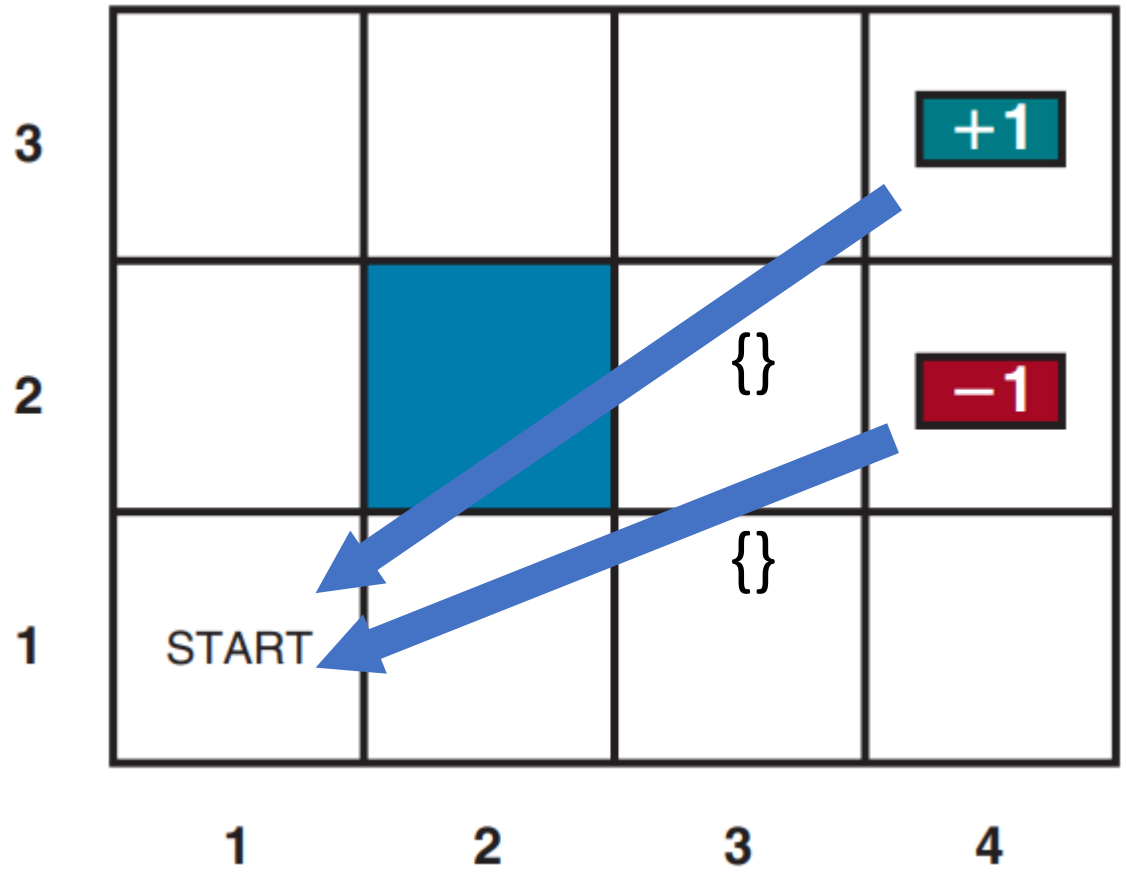
Initial State

Example: Version 2 – Stuck for eternity



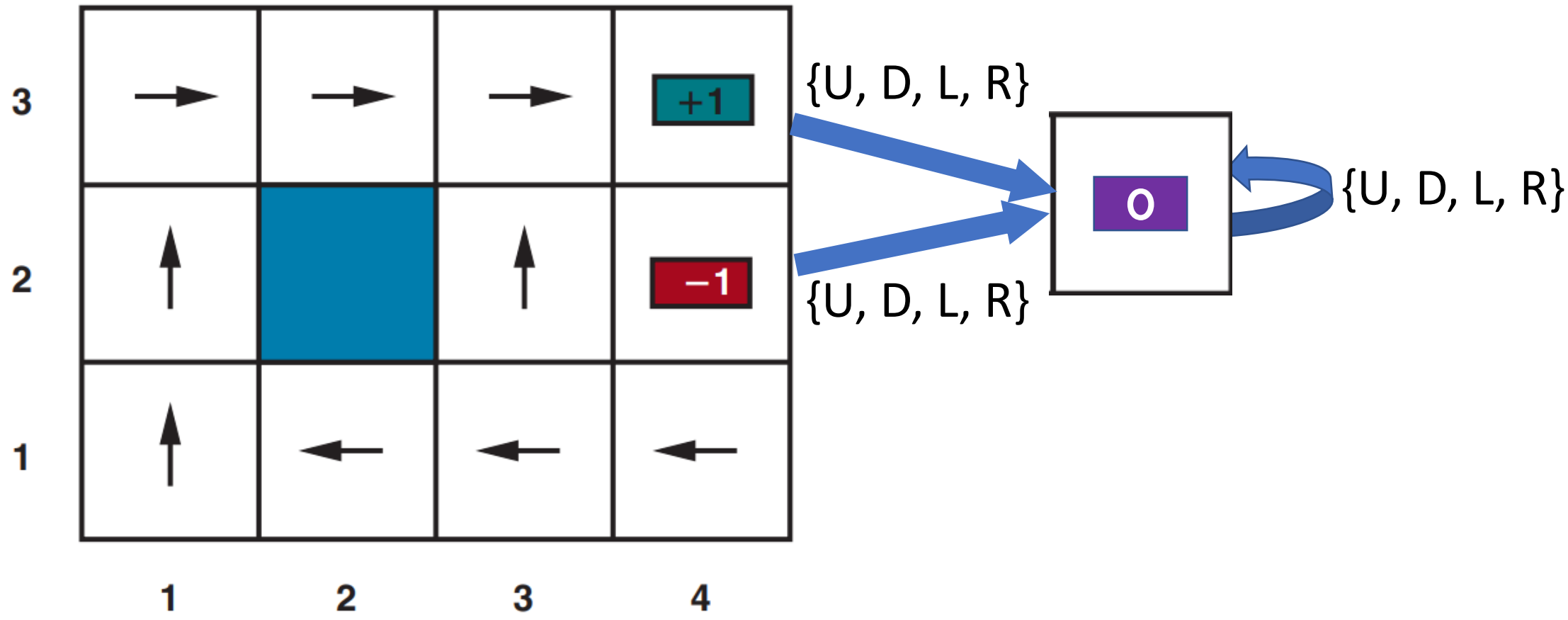
Initial State

Example: Version 3 – Groundhog Day



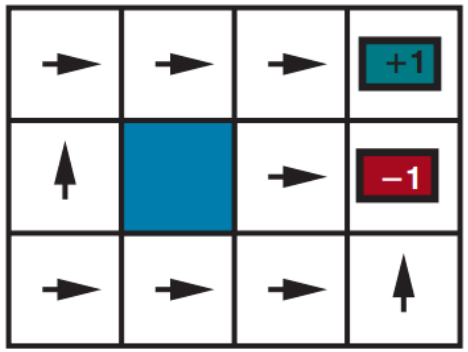
Initial State

Example Policy (Figure 17.2a)

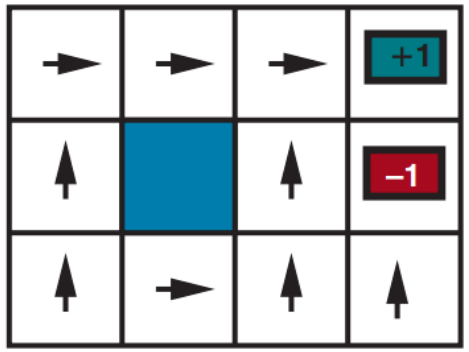


R(s,a,s') = r = -0.04

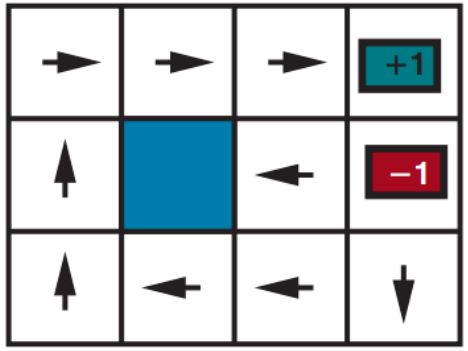
Example Policy (Figure 17.2b)



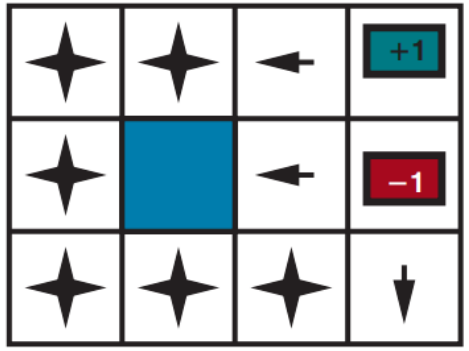
$r < -1.6497$



$-0.7311 < r < -0.4526$

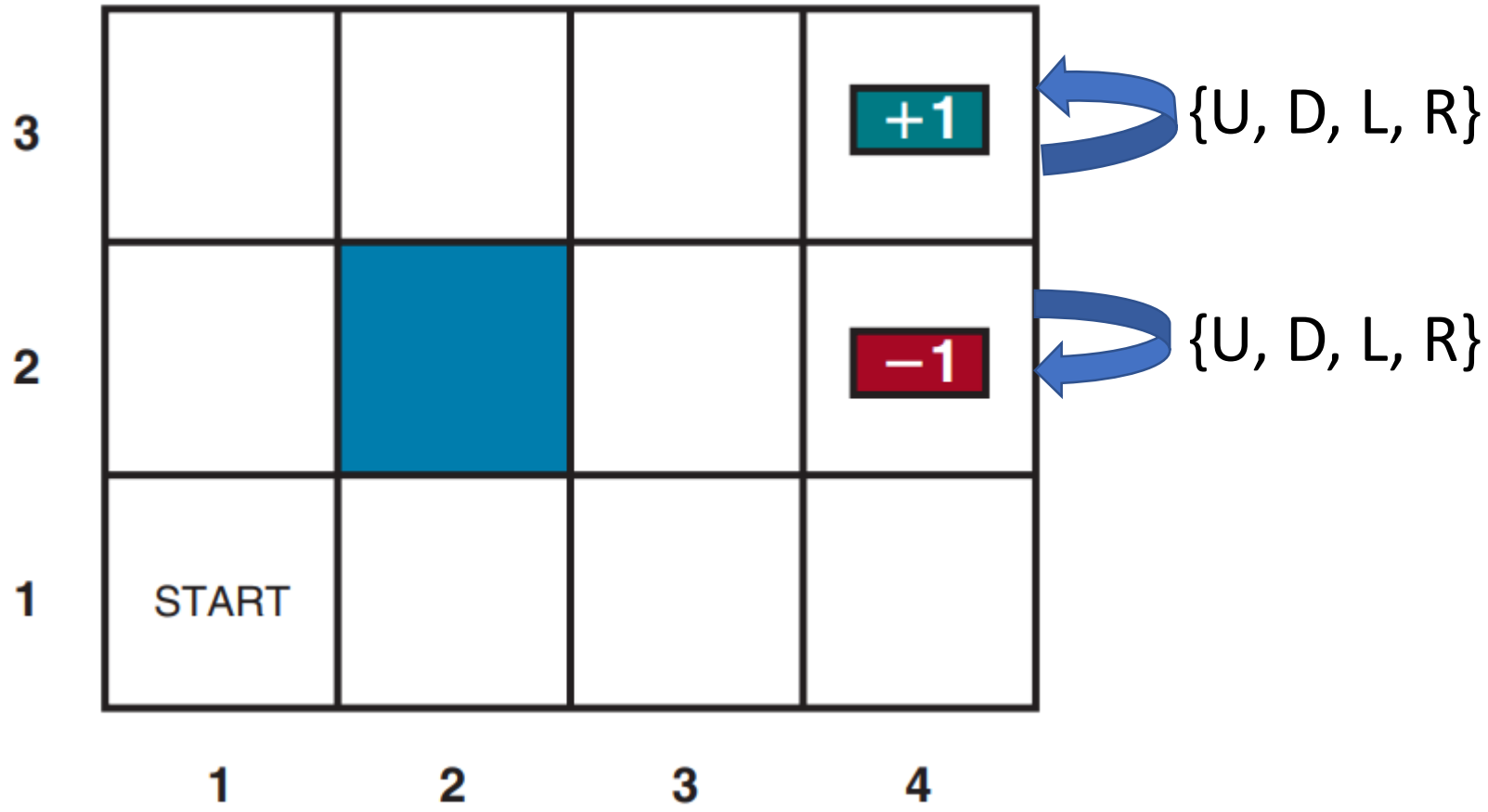


$-0.0274 < r < 0$



$r > 0$

Example: Version 2 – Stuck for eternity



Initial State

What Does “Get Lots of Reward” Mean?

One approach: Cumulative reward

Given: a policy Π (where $a_t = \Pi(s_t), s_{t+1} = \text{apply}(\Pi(s_t), s_t)$)

Cumulative reward:

$$U_C^{\Pi}(s_0) = E\left[\sum_{t=0}^{\infty} \text{R}(s_t, \Pi(s_t), \text{apply}(\Pi(s_t), s_t))\right]$$

Infinite values

What Does “Get Lots of Reward” Mean?

Related approach: Fixed horizon cumulative reward

Given: a policy Π (where $a_t = \Pi(s_t)$, $s_{t+1} = \text{apply}(\Pi(s_t), s_t)$)

Cumulative reward through time *T*:

$$U_T^{\Pi}(s_0) = E\left[\sum_{t=0}^{T} \text{R}(s_t, \Pi(s_t), \text{apply}(\Pi(s_t), s_t))\right]$$

What Does "Get Lots of Reward" Mean?

Our focus: Cumulative discounted reward

Given: a sequence of states $[s_0, s_1, \ldots, s_t, \ldots]$ using actions $[a_0, a_1, \ldots, a_t, \ldots]$

a "discount factor" γ, $0 < \gamma \leq 1$

Cumulative discounted reward:

$$\mathrm{R}(s_0, a_0, s_1) + \gamma\,\mathrm{R}(s_1, a_1, s_2) + \gamma^2\,\mathrm{R}(s_2, a_2, s_3) + \cdots = \sum_{t=0}^{\infty} \gamma^t\,\mathrm{R}(s_t, a_t, s_{t+1})$$

Treats future rewards as exponentially decreasing

What Does "Get Lots of Reward" Mean?

Our focus: Cumulative discounted reward

Given: a sequence of states $[s_0, s_1, \ldots, s_t, \ldots]$ using actions $[a_0, a_1, \ldots, a_t, \ldots]$

a "discount factor" γ, $0 < \gamma \leq 1$

Cumulative discounted reward:

$$\sum_{t=0}^{\infty} \gamma^t \, \mathrm{R}(s_t, a_t, s_{t+1})$$

Treats future rewards as exponentially decreasing

Why a Discount Factor

- Arguments from psychology, economics
- Neurons do something similar
- Math works out

What Does "Get Lots of Reward" Mean?

Our focus: Cumulative discounted reward

Given: a sequence of states $[s_0, s_1, \ldots, s_t, \ldots]$ using actions $[a_0, a_1, \ldots, a_t, \ldots]$

a "discount factor" γ, $0 < \gamma \leq 1$

Cumulative discounted reward:

$$\sum_{t=0}^{\infty} \gamma^t \, \mathrm{R}(s_t, a_t, s_{t+1})$$

Treats future rewards as exponentially decreasing

What Does "Get Lots of Reward" Mean?

Our focus: Cumulative discounted reward

Given: a sequence of states $[s_0, s_1, \ldots, s_t, \ldots]$ using actions $[a_0, a_1, \ldots, a_t, \ldots]$

a "discount factor" γ, $0 < \gamma \leq 1$

Cumulative discounted reward:

$$\sum_{t=0}^{\infty} \gamma^t \, \mathrm{R}(s_t, a_t, s_{t+1})$$

Treats future rewards as exponentially decreasing

t goes to ∞, but converges (if R bounded, $0 \leq \gamma < 1$)

Converges

Imagine $|R(s,a,s')| \leq R_{max}$ for all states and actions

Cumulative discounted reward =

$$\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) \leq \sum_{t=0}^{\infty} \gamma^t R_{max} = R_{max} \sum_{t=0}^{\infty} \gamma^t = \frac{R_{max}}{1-\gamma}$$

Example: If γ=0.5, Cumulative discounted reward $= 2R_{max}$

(similar for negative rewards)

$$a > 1 \qquad 0 < a < 1$$

$$\sum_{i=0}^{n} a^i = \frac{a^{n+1} - 1}{a - 1} = \frac{1 - a^{n+1}}{1 - a}$$

$$\sum_{i=0}^{\infty} a^i = \frac{1}{1 - a}$$

$$0 < a < 1$$

Converges

Imagine $|R(s,a,s')| \leq R_{max}$ for all states and actions

Cumulative discounted reward =

$$\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) \leq \sum_{t=0}^{\infty} \gamma^t R_{max} = R_{max} \sum_{t=0}^{\infty} \gamma^t = \frac{R_{max}}{1-\gamma}$$

Example: If γ=0.5, Cumulative discounted reward $= 2R_{max}$

(similar for negative rewards)

What Does "Get Lots of Reward" Mean?

Our focus: Cumulative discounted reward

Given: a sequence of states [s_0, s_1, …, s_t, …] using actions [a_0, a_1, …, a_t, …]

a "discount factor" γ, $0 \leq \gamma \leq 1$

Cumulative discounted reward:

$$U^{\Pi}(s_0) = \sum_{t=0}^{\infty} \gamma^t \, \mathrm{R}(s_t, a_t, s_{t+1})$$

Treats future rewards as exponentially decreasing

t goes to ∞, but converges (if R bounded, $0 \leq \gamma < 1$)

What Does “Get Lots of Reward” Mean?

Our focus: Cumulative discounted reward

Given: Given: a policy Π (where $a_t = \Pi(s_t)$, $s_{t+1} = \text{apply}(\Pi(s_t), s_t)$)
a “discount factor” γ, 0 ≤ γ ≤ 1

Cumulative discounted reward:

$$U^{\Pi}(s_0) = E\left[\sum_{t=0}^{\infty} \gamma^t \, \text{R}(s_t, \Pi(s_t), \text{apply}(\Pi(s_t), s_t))\right]$$

Definitions

$U^{\Pi}(s)$ tells you what to expect for a particular policy Π starting from state s

Some policy will be better (give bigger $U^{\Pi}(s_0)$) than all others. Call it Π^*.
(There can be multiple such policies - we'll ignore this, doesn't affect anything)

Optimal Policy Π^*: $\Pi^*(s) = \text{argmax}_{\Pi} U^{\Pi}(s)$
Value of optimal policy: $U^{\Pi^*}(s) = U^*(s) = \max_{\Pi} U^{\Pi}(s)$

(book uses U(s) – using * to make optimality clear)

Markov Decision Process

Expands on State Space Search:

- States: S (including some Initial State s_0)
- Actions: A

$$P(s' \mid s, a)\ s \in S,\ a \in A$$

- Reward Function:

$$R: S \times A \times S \rightarrow \mathbb{R}$$

- Goal State/Condition: Find a policy $\Pi*$ that maximizes $U^{\Pi*}(s_0)$

Rewriting Π*

Optimal Policy Π*

$$\mathrm{U}^*(s_0) = \mathrm{argmax}_{\Pi}\mathrm{U}^{\Pi}(s_0) = \mathrm{argmax}_{\Pi}\, E\left[\sum_{t=0}^{\infty} \gamma^t\, \mathrm{R}(s_t, \Pi(s_t), \mathrm{apply}(\Pi(s_t), s_t))\right]$$

Rewriting Π*

Optimal Policy Π*

$$\mathrm{U}^*(s_0) = \mathrm{argmax}_{\Pi}\mathrm{U}^{\Pi}(s_0) = \mathrm{argmax}_{\Pi}\, E\left[\sum_{t=0}^{\infty} \gamma^t\, \mathrm{R}(s_t, \Pi(s_t), \mathrm{apply}(\Pi(s_t), s_t))\right]$$

$$= \underset{a \in A}{\mathrm{argmax}}\, E[\mathrm{R}(s_0, a, \mathrm{apply}(a, s_0)) + \gamma\, \mathrm{U}^{\Pi^*}(\mathrm{apply}(a, s_0))]$$

Intuition

$$U^{\Pi}(s_0) = \sum_{t=0}^{\infty} \gamma^t \, \mathrm{R}(s_t, a_t, s_{t+1}) = \mathrm{R}(s_0, a_0, s_1) + \sum_{t=1}^{\infty} \gamma^t \, \mathrm{R}(s_t, a_t, s_{t+1})$$

$$= \mathrm{R}(s_0, a_0, s_1) + \gamma \sum_{t=1}^{\infty} \gamma^{t-1} \, \mathrm{R}(s_t, a_t, s_{t+1})$$

$$= \mathrm{R}(s_0, a_0, s_1) + \gamma \sum_{t=0}^{\infty} \gamma^{t} \, \mathrm{R}(s_{t+1}, a_{t+1}, s_{t+2})$$

$$= \mathrm{R}(s_0, a_0, s_1) + \gamma[\mathrm{R}(s_1, a_1, s_2) + \gamma \, \mathrm{R}(s_2, a_2, s_3) + \gamma^2 \, \mathrm{R}(s_3, a_3, s_4) + \cdots]$$

Intuition

$$U^{\Pi}(s_0) = \sum_{t=0}^{\infty} \gamma^t \, \mathrm{R}(s_t, a_t, s_{t+1}) = \mathrm{R}(s_0, a_0, s_1) + \sum_{t=1}^{\infty} \gamma^t \, \mathrm{R}(s_t, a_t, s_{t+1})$$

$$= \mathrm{R}(s_0, a_0, s_1) + \gamma \sum_{t=1}^{\infty} \gamma^{t-1} \, \mathrm{R}(s_t, a_t, s_{t+1})$$

$$= \mathrm{R}(s_0, a_0, s_1) + \gamma \sum_{t=0}^{\infty} \gamma^{t} \, \mathrm{R}(s_{t+1}, a_{t+1}, s_{t+2})$$

$$= \mathrm{R}(s_0, a_0, s_1) + \gamma [\mathrm{R}(s_1, a_1, s_2) + \gamma \, \mathrm{R}(s_2, a_2, s_3) + \gamma^2 \, \mathrm{R}(s_3, a_3, s_4) + \cdots]$$

$$U^{\Pi}(s_1)$$

Intuition

$$U^{\Pi}(s_0) = \mathrm{R}(s_0, a_0, s_1) + \gamma\, U^{\Pi}(s_1)$$

Rewriting Π*

Value of optimal Policy Π*

$$U^*(s_0) = \text{argmax}_\Pi U^\Pi(s_0) = \text{argmax}_\Pi E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \Pi(s_t), \text{apply}(\Pi(s_t), s_t))\right]$$

$$= \underset{a \in A}{\text{argmax}}\, E[R(s_0, a, \text{apply}(a, s_0)) + \gamma\, U^{\Pi^*}(\text{apply}(a, s_0))]$$

Rewriting Π*

Value of optimal Policy Π*

$$\mathrm{U}^*(s_0) = \mathrm{argmax}_{\Pi}\mathrm{U}^{\Pi}(s_0) = \mathrm{argmax}_{\Pi} E\left[\sum_{t=0}^{\infty} \gamma^t \mathrm{R}(s_t, \Pi(s_t), \mathrm{apply}(\Pi(s_t), s_t))\right]$$

$$= \underset{a \in A}{\mathrm{argmax}}\, E[\mathrm{R}(s_0, a, \mathrm{apply}(a, s_0)) + \gamma\, \mathrm{U}^{\Pi^*}(\mathrm{apply}(a, s_0))]$$

$$= \underset{a \in A}{\mathrm{argmax}} \sum_{s' \in S} \mathrm{P}(s'|s_0,a)[\mathrm{R}(s_0, a, s') + \gamma\, \mathrm{U}^{\Pi^*}(s')]$$

Rewriting U*

Value of optimal Policy U*

$$\mathrm{U}^*(s_0) = \max_{\Pi} \mathrm{U}^{\Pi}(s_0) = \max_{\Pi} E\left[\sum_{t=0}^{\infty} \gamma^t \, \mathrm{R}(s_t, \Pi(s_t), \mathrm{apply}(\Pi(s_t), s_t)))\right]$$

$$= \max_{a \in A} E[\mathrm{R}(s_0, a, \mathrm{apply}(a, s_0)) + \gamma \, \mathrm{U}^{\Pi *}(\mathrm{apply}(a, s_0))]$$

$$= \max_{a \in A} \sum_{s' \in S} \mathrm{P}(s'|s_0,\mathrm{a})[\mathrm{R}(s_0, a, s') + \gamma \, \mathrm{U}^{\Pi *}(s')]$$

Known as the “Bellman Equation”

Rewriting U*

Value of optimal Policy U*

$$U^*(s_0) = \max_{\Pi} U^{\Pi}(s_0) = \max_{\Pi} E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \Pi(s_t), \text{apply}(\Pi(s_t), s_t))\right]$$

$$= \max_{a \in A} E[R(s_0, a, \text{apply}(a, s_0)) + \gamma U^{\Pi*}(\text{apply}(a, s_0))]$$

$$= \max_{a \in A} \sum_{s' \in S} P(s'|s_0, a)[R(s_0, a, s') + \gamma U^{\Pi*}(s')]$$

Known as the “Bellman Equation”

Rewriting U^{Π}

$$U^{\Pi}(s_0) = E\left[\sum_{t=0}^{\infty} \gamma^t \, \mathrm{R}(s_t, \Pi(s_t), \mathrm{apply}(\Pi(s_t), s_t)))\right]$$

$$= E[\mathrm{R}(s_0, \Pi(s_t), \mathrm{apply}(\Pi(s_t), s_0)) + \gamma \, U^{\Pi*}(\mathrm{apply}(\Pi(s_t), s_0))]$$

$$= \sum_{s' \in S} \mathrm{P}(s'|s, \Pi(s))[\mathrm{R}(s, \Pi(s), s') + \gamma \, U^{\Pi}(s')]$$

MDPs and Reinforcement Learning: Game Plan

- Figure out Π* if you know S, R, and P – no learning
- Figure out Π* if you know S and P but not R – learning
- Figure out Π* if you know S but not R and P – learning

(|S| finite and ~small)

MDPs and Reinforcement Learning: Game Plan

- Figure out Π* if you know S, R, and P – no learning
- Figure out Π* if you know S and P but not R – learning
- Figure out Π* if you know S but not R and P – learning

(|S| finite and ~small)

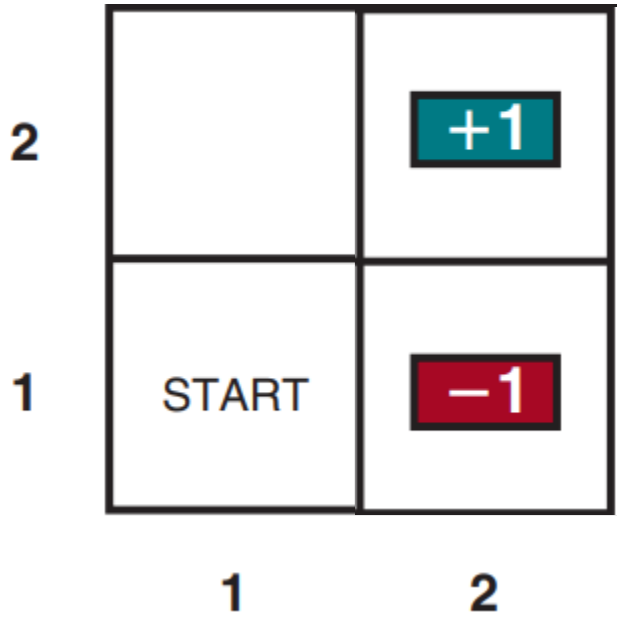
First:

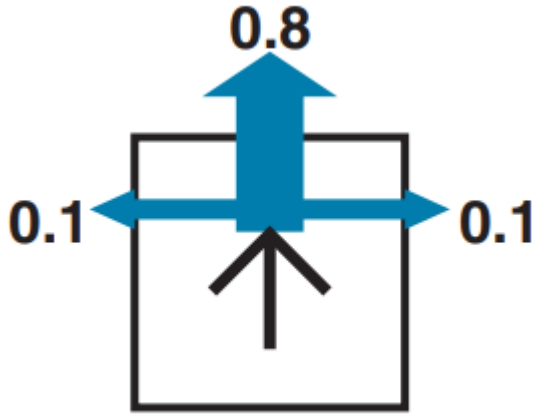
How can you compute U^{Π}

for a specific policy Π (not necessarily optimal)

if you know P and R?

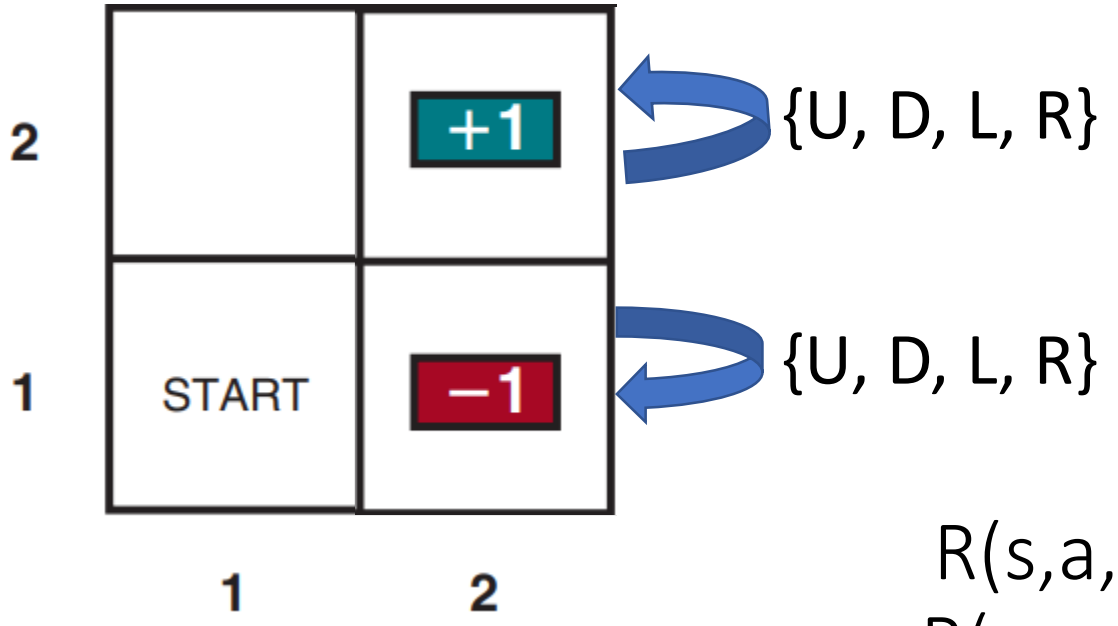
Simplified Example





R(s,a,<2,2>) = 1
R(s,a,<2,1>) = -1
R(s,a,s') = r = -.04
for all other states

Simplified Example – Version 2



R(s,a,<2,2>) = 1
R(s,a,<2,1>) = -1
R(s,a,s') = r = -.04
for all other states

Example

Policy Π_{UR} = Go up if you're in row 1, otherwise go right

Approach 1: Can compute from the definitions above

Thus for example $U^{\Pi_{UR}}(<1,1>)$ represents what your expected cumulative reward will be if you start at <1,1> and select an action at each step using Π_{UR} – always going up in row 1, right in row 2.

Example

Policy Π_{UR} = Go up if you're in row 1, otherwise go right

$$U^{\Pi_{UR}}(<1,1>) = 0.8 \times [R(<1,1>,U,<1,2>) + \gamma \times U^{\Pi_{UR}}(<1,2>)]$$
$$+ 0.1 \times [R(<1,1>,U,<1,1>) + \gamma \times U^{\Pi_{UR}}(<1,1>)]$$
$$+ 0.1 \times [R(<1,1>,U,<2,1>) + \gamma \times U^{\Pi_{UR}}(<2,1>)]$$

This uses the definitions of U from the previous slides together with the definition of actions, P, and R

Example

Policy Π_{UR} = Go up if you're in row 1, otherwise go right

$$U^{\Pi_{UR}}(<1,1>) = 0.8 \times [R(<1,1>,U,<1,2>) + \gamma \times U^{\Pi_{UR}}(<1,2>)]$$
$$+ 0.1 \times [R(<1,1>,U,<1,1>) + \gamma \times U^{\Pi_{UR}}(<1,1>)]$$
$$+ 0.1 \times [R(<1,1>,U,<2,1>) + \gamma \times U^{\Pi_{UR}}(<2,1>)]$$
$$= 0.8 \times [-0.04 + \gamma \times U^{\Pi_{UR}}(<1,2>)]$$
$$+ 0.1 \times [-0.04 + \gamma \times U^{\Pi_{UR}}(<1,1>)]$$
$$+ 0.1 \times [-0.04 + \gamma \times U^{\Pi_{UR}}(<2,1>)]$$
$$= -0.04 + \gamma \times [0.8 \times U^{\Pi_{UR}}(<1,2>) + 0.1 \times U^{\Pi_{UR}}(<1,1>) + 0.1 \times U^{\Pi_{UR}}(<2,1>)]$$

Example

Policy Π_{UR} = Go up if you're in row 1, otherwise go right

$$U^{\Pi_{UR}}(<1,1>) = 0.8 \times [R(<1,1>,U,<1,2>) + \gamma \times U^{\Pi_{UR}}(<1,2>)]$$
$$+ 0.1 \times [R(<1,1>,U,<1,1>) + \gamma \times U^{\Pi_{UR}}(<1,1>)]$$
$$+ 0.1 \times [R(<1,1>,U,<2,1>) + \gamma \times U^{\Pi_{UR}}(<2,1>)]$$
$$= 0.8 \times [-0.04 + \gamma \times U^{\Pi_{UR}}(<1,2>)]$$
$$+ 0.1 \times [-0.04 + \gamma \times U^{\Pi_{UR}}(<1,1>)]$$
$$+ 0.1 \times [-0.04 + \gamma \times U^{\Pi_{UR}}(<2,1>)]$$
$$= -0.04 + \gamma \times [0.8 \times U^{\Pi_{UR}}(<1,2>) + 0.1 \times U^{\Pi_{UR}}(<1,1>) + 0.1 \times U^{\Pi_{UR}}(<2,1>)]$$

And similarly for <1,2>, <2,1>, and <2,2>

Example

Policy Π_{UR} = Go up if you're in row 1, otherwise go right

$U^{\Pi_{UR}}(<1,1>) = -0.04 + \gamma \times [0.8 \times U^{\Pi_{UR}}(<1,2>) + 0.1 \times U^{\Pi_{UR}}(<1,1>) + 0.1 \times U^{\Pi_{UR}}(<2,1>)]$

$U^{\Pi_{UR}}(<1,2>) = -0.04 + \gamma \times [0.8 \times U^{\Pi_{UR}}(<2,2>) + 0.1 \times U^{\Pi_{UR}}(<1,2>) + 0.1 \times U^{\Pi_{UR}}(<1,1>)]$

$U^{\Pi_{UR}}(<2,1>) = -1.0 + \gamma \times [0.8 \times U^{\Pi_{UR}}(<2,2>) + 0.1 \times U^{\Pi_{UR}}(<1,1>) + 0.1 \times U^{\Pi_{UR}}(<2,1>)]$

$U^{\Pi_{UR}}(<2,2>) = 1.0 + \gamma \times [0.8 \times U^{\Pi_{UR}}(<2,2>) + 0.1 \times U^{\Pi_{UR}}(<2,2>) + 0.1 \times U^{\Pi_{UR}}(<2,1>)]$

Example

Policy Π_{UR} = Go up if you're in row 1, otherwise go right

$$U^{\Pi_{UR}}(<1,1>) = -0.04 + \gamma \times [0.8 \times U^{\Pi_{UR}}(<1,2>) + 0.1 \times U^{\Pi_{UR}}(<1,1>) + 0.1 \times U^{\Pi_{UR}}(<2,1>)]$$

$$U^{\Pi_{UR}}(<1,2>) = -0.04 + \gamma \times [0.8 \times U^{\Pi_{UR}}(<2,2>) + 0.1 \times U^{\Pi_{UR}}(<1,2>) + 0.1 \times U^{\Pi_{UR}}(<1,1>)]$$

$$U^{\Pi_{UR}}(<2,1>) = -1.0 + \gamma \times [0.8 \times U^{\Pi_{UR}}(<2,2>) + 0.1 \times U^{\Pi_{UR}}(<1,1>) + 0.1 \times U^{\Pi_{UR}}(<2,1>)]$$

$$U^{\Pi_{UR}}(<2,2>) = 1.0 + \gamma \times [0.8 \times U^{\Pi_{UR}}(<2,2>) + 0.1 \times U^{\Pi_{UR}}(<2,2>) + 0.1 \times U^{\Pi_{UR}}(<2,1>)]$$

Four equations, four unknowns, for a given γ can now solve
(for example, with Gaussian Elimination)

MDPs and Reinforcement Learning: Game Plan

- Figure out Π* if you know S, R, and P – no learning
- Figure out Π* if you know S and P but not R – learning
- Figure out Π* if you know S but not R and P – learning

(|S| finite and ~small)

$$U^*(<1,2>) = \max_{a \in A} \{0.8 \times [-0.04 + \gamma U^*(<1,2>)] + 0.1 \times [-0.04 + \gamma U^*(<1,2>)] + 0.1 \times [-0.04 + \gamma U^*(<2,2>)] \quad \text{UP}$$

$$0.8 \times [-0.04 + \gamma U^*(<1,1>)] + 0.1 \times [-0.04 + \gamma U^*(<1,2>)] + 0.1 \times [-0.04 + \gamma U^*(<2,2>)] \quad \text{DOWN}$$

$$0.8 \times [-0.04 + \gamma U^*(<1,2>)] + 0.1 \times [-0.04 + \gamma U^*(<1,2>)] + 0.1 \times [-0.04 + \gamma U^*(<1,1>)] \quad \text{LEFT}$$

$$0.8 \times [-0.04 + \gamma U^*(<2,2>)] + 0.1 \times [-0.04 + \gamma U^*(<1,2>)] + 0.1 \times [-0.04 + \gamma U^*(<1,1>)]\} \quad \text{RIGHT}$$

$$\Pi^*(<1,2>) = \text{argmax}_{a \in A} \{0.8 \times [-0.04 + \gamma U^*(<1,2>)] + 0.1 \times [-0.04 + \gamma U^*(<1,2>)] + 0.1 \times [-0.04 + \gamma U^*(<2,2>)] \quad \text{UP}$$

$$0.8 \times [-0.04 + \gamma U^*(<1,1>)] + 0.1 \times [-0.04 + \gamma U^*(<1,2>)] + 0.1 \times [-0.04 + \gamma U^*(<2,2>)] \quad \text{DOWN}$$

$$0.8 \times [-0.04 + \gamma U^*(<1,2>)] + 0.1 \times [-0.04 + \gamma U^*(<1,2>)] + 0.1 \times [-0.04 + \gamma U^*(<1,1>)] \quad \text{LEFT}$$

$$0.8 \times [-0.04 + \gamma U^*(<2,2>)] + 0.1 \times [-0.04 + \gamma U^*(<1,2>)] + 0.1 \times [-0.04 + \gamma U^*(<1,1>)]\} \quad \text{RIGHT}$$

$$U^*(<1,2>) = \max_{a \in A} \{0.8 \times [-0.04 + \gamma\, U^*(<1,2>)] + 0.1 \times [-0.04 + \gamma\, U^*(<1,2>)] + 0.1 \times [-0.04 + \gamma\, U^*(<2,2>)] \quad \text{UP}$$

$$0.8 \times [-0.04 + \gamma\, U^*(<1,1>)] + 0.1 \times [-0.04 + \gamma\, U^*(<1,2>)] + 0.1 \times [-0.04 + \gamma\, U^*(<2,2>)] \quad \text{DOWN}$$

$$0.8 \times [-0.04 + \gamma\, U^*(<1,2>)] + 0.1 \times [-0.04 + \gamma\, U^*(<1,2>)] + 0.1 \times [-0.04 + \gamma\, U^*(<1,1>)] \quad \text{LEFT}$$

$$0.8 \times [-0.04 + \gamma\, U^*(<2,2>)] + 0.1 \times [-0.04 + \gamma\, U^*(<1,2>)] + 0.1 \times [-0.04 + \gamma\, U^*(<1,1>)]\} \quad \text{RIGHT}$$

$$\Pi^*(<1,2>) = \text{argmax}_{a \in A} \{0.8 \times [-0.04 + \gamma\, U^*(<1,2>)] + 0.1 \times [-0.04 + \gamma\, U^*(<1,2>)] + 0.1 \times [-0.04 + \gamma\, U^*(<2,2>)] \quad \text{UP}$$

$$0.8 \times [-0.04 + \gamma\, U^*(<1,1>)] + 0.1 \times [-0.04 + \gamma\, U^*(<1,2>)] + 0.1 \times [-0.04 + \gamma\, U^*(<2,2>)] \quad \text{DOWN}$$

$$0.8 \times [-0.04 + \gamma\, U^*(<1,2>)] + 0.1 \times [-0.04 + \gamma\, U^*(<1,2>)] + 0.1 \times [-0.04 + \gamma\, U^*(<1,1>)] \quad \text{LEFT}$$

$$0.8 \times [-0.04 + \gamma\, U^*(<2,2>)] + 0.1 \times [-0.04 + \gamma\, U^*(<1,2>)] + 0.1 \times [-0.04 + \gamma\, U^*(<1,1>)]\} \quad \text{RIGHT}$$

If you write down U* for every state it no longer gives a set of linear equations to which you can apply Gaussian elimination

If you write down U* for every state it no longer gives a set of linear equations that you can apply Gaussian elimination to

What to do?

$$\text{Policy } \Pi^* = \text{argmax}_{\Pi} U^{\Pi}$$

$$\text{Policy } \Pi^* = \text{argmax}_{\Pi} U^{\Pi}$$

Can we enumerate all policies to find the best one?

Policy $\Pi^* = \text{argmax}_{\Pi} U^{\Pi}$

Can we enumerate all policies to find the best one?

Rarely

It’s usually intractable

Policy $\Pi^* = \text{argmax}_{\Pi} U^{\Pi}$

Can we enumerate all policies to find the best one?

Rarely

It’s usually intractable

What can we do?

What to do?

Policy Iteration algorithm
(among various algorithms for doing this)

Intuition

How might I compute Π*(s) if I knew U*(s)?

If I know the expected values, would that help me figure out what action to take?

U*(s) just tells me the expected cumulative discounted reward, not what action to take…

U*(s) just tells me the expected cumulative discounted reward, not what action to take…

$$\Pi^*(s) = \text{argmax}_{a \in A} \sum_{s' \in S} P(s'|s,a)[R(s,a,s') + \gamma U^*(s')]$$

Look at each action, figure out which gives me the best expected value
I can compute this if I know U*

Finding Π*: Policy Iteration

Notation:

- $\hat{\pi}$: Evolving approximation of policy
- $\hat{\pi}_i$: Approximation on iteration I
- $\hat{U}$: Evolving approximation of U
- $\hat{U}_i$: Approximation on iteration i

Finding Π^*: Policy Iteration

Policy_Iteration(S,A,P,R,γ):

For all s$\in$S { $\hat{\pi}_0(s) \leftarrow$ <random a$\in$A>; $\hat{U}_0(s) \leftarrow 0$; i $\leftarrow$ 0 }

Repeat

i $\leftarrow$ i+1

For all s$\in$S

$$\hat{U}_i(s) \leftarrow R(s_t, a_t, s_{t+1}) + \gamma \sum_{s' \in S} [P(s' | s, \hat{\pi}_{i-1}(s))\, \hat{U}_{i-1}(s')]$$

For all s$\in$S

If $\max_{a \in A} \sum_{s' \in S} [P(s' | s, a)\, \hat{U}_i(s')] > \sum_{s' \in S} [P(s' | s, \hat{\pi}_{i-1}(s))\, \hat{U}_i(s')]$

Then $\hat{\pi}_i(s) \leftarrow \operatorname{argmax}_{a \in A} \sum_{s' \in S} [P(s' | s, a)\, \hat{U}_i(s')]$

Else $\hat{\pi}_i(s) \leftarrow \hat{\pi}_{i-1}(s)$

Until <stopping condition> /* for example: $\hat{\pi}$ doesn't change */