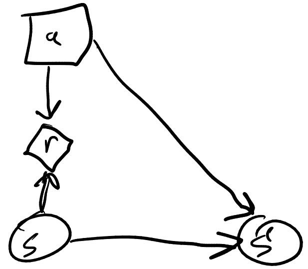
Policy and Value Iteration

Last Time

(S, A, R, T, γ)

Last Time

- How is a **Markov decision process** defined?



Last Time

$a \leftarrow \pi(s)$

- How is a **Markov decision process** defined?
- What is a **policy**?

Last Time

- How is a **Markov decision process** defined?
- What is a **policy**?
- How do we **evaluate** policies?

Monte Carlo

Value - based

Last Time

- How is a **Markov decision process** defined?
- What is a **policy**?
- How do we **evaluate** policies?

(MDP notebook)

for initial state distribution b

Want $U(\pi) = \sum_{s \in S} b(s)\, U^\pi(s)$

$U^\pi(s) = R(s, \pi(s)) + \gamma \sum_{s'} T(s' \mid s, a)\, U^\pi(s')$ Bellman Expectation Equation

$\vec{U}^\pi = (I - \gamma T^\pi)^{-1} \vec{R}^\pi$

$\uparrow$

$\vec{U}^\pi[i] = U^\pi(i)$

Last Time

$T(s' \mid s, a)$

$s', r \leftarrow G(s, a)$

- How is a **Markov decision process** defined?
- What is a **policy**?
- How do we **evaluate** policies?

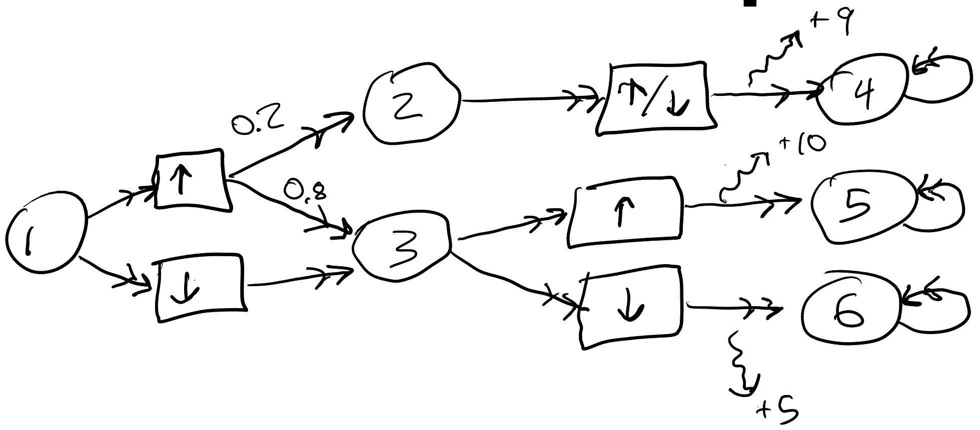
(MDP notebook)

Guiding Questions

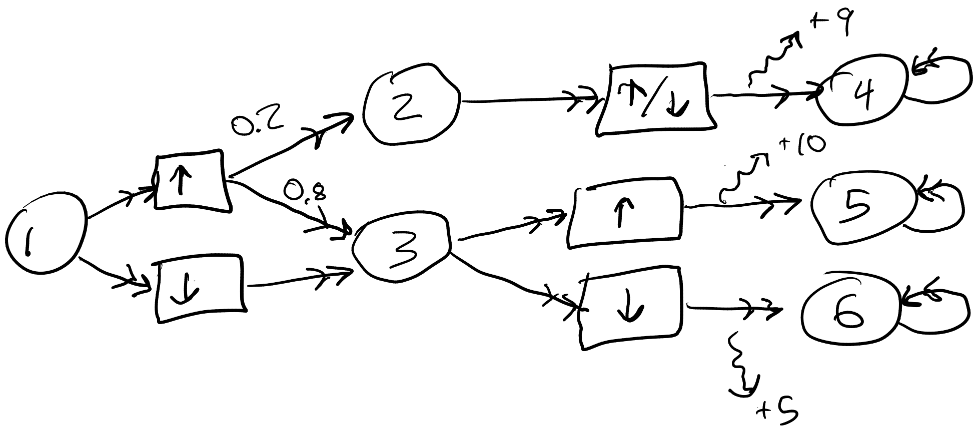
Guiding Questions

- How do we reason about the **future consequences** of actions in an MDP?
- What are the basic **algorithms for solving MDPs**?

MDP Example: Up-Down Problem

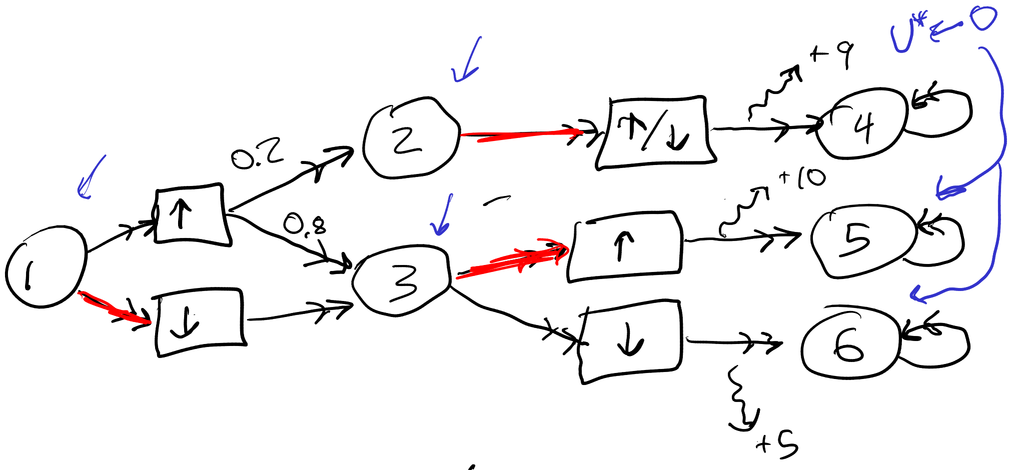


Dynamic Programming and Value Backup



Dynamic Programming and Value Backup

$\gamma = 1$



Bellman Backup

$U^*(s) \leftarrow 0$ for all terminal states

Repeat until all $U^*(s)$ is calculated:

find π^*, U^* for all states where U^* is known for all successor states

$\pi^*(s)$

Bellman's Principle of Optimality: Every sub-policy in an optimal policy is locally optimal

$$U^*(s) = \max_{\pi} U^{\pi}(s)$$

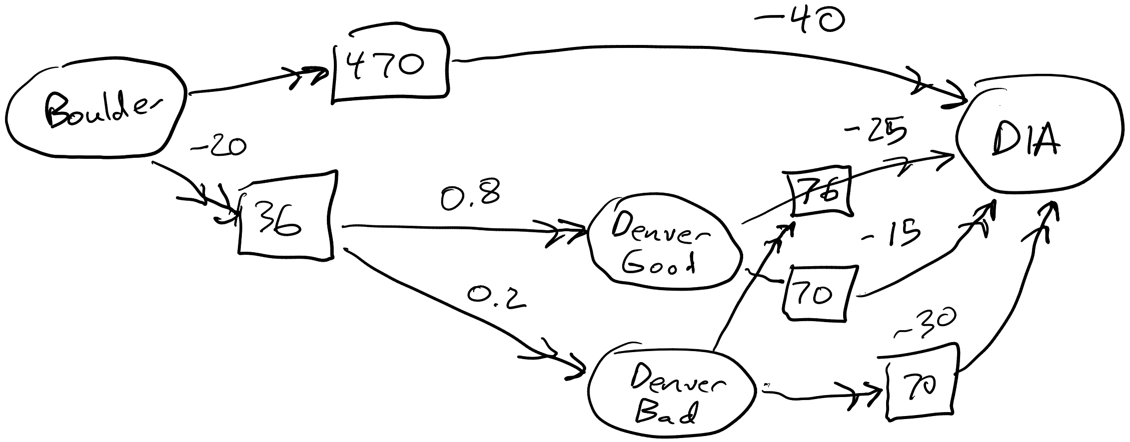
1 step

$$\pi^*(s) = \underset{a \in A}{\operatorname{argmax}} \left(\underbrace{R(s,a) + \gamma E[U^*(s')]}_{Q^*(s,a)} \right)$$

$$U^*(s) = \max_a Q^*(s,a)$$

s	$U^*(s)$	a	$Q^*(s,a)$
4	0		
5	0		
6	0		
3	+10	$\boxed{\uparrow}$ ↓	$R(3,\uparrow) + \gamma U^*(5) = +10$ $R(3,\downarrow) + \gamma U^*(6) = +5$
2	+9	$\boxed{\uparrow\downarrow}$	$R(2,\cdot) + \gamma U^*(4) = +9$
1	+10	↑ $\boxed{\downarrow}$	$R(1,\uparrow) + \gamma(0.2\,U^*(2) + 0.8\,U^*(3)) = 9.8$ $R(1,\downarrow) + \gamma U^*(3) = 10$

Break: DIA Run



$\boxed{36}$

$37 = U^*(\text{Boulder})$

Policy Iteration

Algorithm: Policy Iteration

Given: MDP (S, A, R, T, γ, b)

Policy Iteration

Algorithm: Policy Iteration

Given: MDP (S, A, R, T, γ, b)

1. initialize π, π' (differently)

Policy Iteration

Algorithm: Policy Iteration

Given: MDP (S, A, R, T, γ, b)

1. initialize π, π' (differently)
2. while $\pi \neq \pi'$

Policy Iteration

Algorithm: Policy Iteration

Given: MDP (S, A, R, T, γ, b)

1. initialize π, π' (differently)
2. while $\pi \neq \pi'$
3. $\quad \pi \leftarrow \pi'$

Policy Iteration

Algorithm: Policy Iteration

Given: MDP (S, A, R, T, γ, b)

1. initialize π, π' (differently)
2. while $\pi \neq \pi'$
3. $\quad \pi \leftarrow \pi'$
4. $\quad U^\pi \leftarrow (I - \gamma T^\pi)^{-1} R^\pi$

Policy Iteration

Algorithm: Policy Iteration

Given: MDP (S, A, R, T, γ, b)

1. initialize π, π' (differently)
2. while $\pi \neq \pi'$
3. $\quad \pi \leftarrow \pi'$
4. $\quad U^\pi \leftarrow (I - \gamma T^\pi)^{-1} R^\pi$
5. $\quad \pi'(s) \leftarrow \underset{a \in A}{\text{argmax}} \left(R(s,a) + \gamma \sum_{s' \in S} T(s'|s,a) U^\pi(s')\right) \quad \forall s \in S$

Policy Iteration

Algorithm: Policy Iteration

Given: MDP (S, A, R, T, γ, b)

1. initialize π, π' (differently)
2. while $\pi \neq \pi'$
3. $\quad \pi \leftarrow \pi'$
4. $\quad U^\pi \leftarrow (I - \gamma T^\pi)^{-1} R^\pi$ Evaluation
5. $\quad \pi'(s) \leftarrow \underset{a \in A}{\text{argmax}} \left(R(s,a) + \gamma \sum_{s' \in S} T(s' | s, a) U^\pi(s') \right) \quad \forall s \in S$ Update
6. return π

Policy Iteration

Algorithm: Policy Iteration

Given: MDP (S, A, R, T, γ, b)

1. initialize π, π' (differently)
2. while $\pi \neq \pi'$
3. $\quad \pi \leftarrow \pi'$
4. $\quad U^\pi \leftarrow (I - \gamma T^\pi)^{-1} R^\pi$
5. $\quad \pi'(s) \leftarrow \underset{a \in A}{\text{argmax}} \left(R(s,a) + \gamma \sum_{s' \in S} T(s'|s,a) U^\pi(s') \right) \quad \forall s \in S$
6. return π

(Policy iteration notebook)

Value Iteration

Algorithm: Value Iteration

Given: MDP (S, A, R, T, γ, b), tolerance ϵ

Value Iteration

<u>Algorithm: Value Iteration</u>

Given: MDP (S, A, R, T, γ, b), tolerance ϵ

1. initialize U, U' (differently)

Value Iteration

Algorithm: Value Iteration

Given: MDP (S, A, R, T, γ, b), tolerance ϵ

1. initialize U, U' (differently)
2. while $\|U - U'\|_\infty > \epsilon$

Value Iteration

Algorithm: Value Iteration

Given: MDP (S, A, R, T, γ, b), tolerance ϵ

1. initialize U, U' (differently)
2. while $\|U - U'\|_\infty > \epsilon$
3. $\quad U \leftarrow U'$

Value Iteration

<u>Algorithm: Value Iteration</u>

Given: MDP (S, A, R, T, γ, b), tolerance ϵ

1. initialize U, U' (differently)
2. while $\|U - U'\|_\infty > \epsilon$
3. $\quad U \leftarrow U'$
4. $\quad U'(s) \leftarrow \max\limits_{a \in A} \left(R(s,a) + \gamma \sum_{s' \in S} T(s'|s,a) U(s')\right) \qquad \forall s \in S$

Value Iteration

<u>Algorithm: Value Iteration</u>

Given: MDP (S, A, R, T, γ, b), tolerance ϵ

1. initialize U, U' (differently)
2. while $\|U - U'\|_\infty > \epsilon$
3. $\quad U \leftarrow U'$
4. $\quad U'(s) \leftarrow \max\limits_{a \in A} \left(R(s,a) + \gamma \sum_{s' \in S} T(s'|s,a) U(s')\right) \quad \forall s \in S$
5. return U'

Value Iteration

Algorithm: Value Iteration

Given: MDP (S, A, R, T, γ, b), tolerance ϵ

1. initialize U, U' (differently)
2. while $\|U - U'\|_\infty > \epsilon$
3. $\quad U \leftarrow U'$
4. $\quad U'(s) \leftarrow \max_{a \in A} \left(R(s,a) + \gamma \sum_{s' \in S} T(s'|s,a) U(s')\right) \quad \forall s \in S$
5. return U'

- Returned U' will be close to U^*!

Value Iteration

Algorithm: Value Iteration

Given: MDP (S, A, R, T, γ, b), tolerance ϵ

1. initialize U, U' (differently)
2. while $\|U - U'\|_\infty > \epsilon$
3. $\quad U \leftarrow U'$
4. $\quad U'(s) \leftarrow \max_{a \in A} \left(R(s,a) + \gamma \sum_{s' \in S} T(s'|s,a) U(s')\right) \quad \forall s \in S$
5. return U'

- Returned U' will be close to U^*!
- π^* is easy to extract: $\pi^*(s) = \arg\max(R(s,a) + \gamma E[U^*(s)])$

Bellman's Equations

Policy Evaluation

$$U^{\pi}(s) = R(s, \pi(s)) + \gamma \underset{s' \sim T(s,\pi(s))}{E}\left[U^{\pi}(s')\right]$$

Bellman's Expectation Equation

Bellman Backup

Certificate of Optimality

$$U^{*}(s) = \max_a \left(R(s,a) + \gamma \underset{s' \sim T(s,a)}{E}\left[U^{*}(s')\right]\right)$$

Bellman's Optimality Equation

Value Iteration

$$U'(s) = \max_a \left(R(s,a) + \gamma \underset{s' \sim T(s,a)}{E}\left[U(s')\right]\right)$$

Bellman Operator

$$U'(s) = B[U](s) = \text{''} \qquad \text{''}$$

VI

initialize U, U'

while $\|U - U'\|_\infty > \varepsilon$

$U \leftarrow U'$

$U' \leftarrow B[U]$

Guiding Questions

Guiding Questions

- How do we reason about the **future consequences** of actions in an MDP?
- What are the basic **algorithms for solving MDPs**?

Guiding Questions

- How do we reason about the **future consequences** of actions in an MDP?
- What are the basic **algorithms for solving MDPs**?

"In any small change he will have to consider only these quantitative indices (or "values") in which all the relevant information is concentrated; and by adjusting the quantities one by one, he can appropriately rearrange his dispositions without having to solve the whole puzzle ab initio, or without needing at any stage to survey it at once in all its ramifications."

-- F. A. Hayek, "The use of knowledge in society", 1945