Artificial Intelligence

Planning 6: Solving MDPs

Jendrik Seipp

Linköping University

based on slides by Thomas Keller and Malte Helmert (University of Basel)

Introduction

Computation of an Optimal Policy

there are various techniques to compute an optimal policy:

- (linear program)
- policy iteration
- value iteration

Computation of an Optimal Policy

there are various techniques to compute an optimal policy:

- (linear program)
- policy iteration
- value iteration

policy iteration consists of 2 components:

- policy evaluation: compute V_π for a given π
- policy improvement: use V_π to determine better policy

Policy Evaluation

Policy Evaluation: Implementations

computing the state value of all states for a given policy π (i.e., computing V_π) is called policy evaluation

there are several algorithms for policy evaluation:

1. linear program
2. backward induction
3. iterative policy evaluation

Iterative Policy Evaluation: Idea

- impossible to compute state-values
  in one sweep over the state space in presence of cycles
- start with arbitrary state-value function $\hat{V}_\pi^0$
- treat state-value function as update rule

$$\hat{V}_\pi^i(s) = \sum_{s' \in \text{succ}(s,\pi(s))} T(s,\pi(s),s') \cdot \Big(R(s,\pi(s),s') + \gamma \cdot V_\pi^{i-1}(s')\Big)$$

- apply update rule iteratively
- until state-values have converged
  (in practice: until maximal change is smaller than residual ϵ)

Iterative Policy Evaluation: Example

	1	2	3	4
5	$\Rightarrow$ 0.00	$\Rightarrow$ 0.00	$\Rightarrow$ 0.00	noop 0.00
4	$\Rightarrow$ 0.00	$\Uparrow$ 0.00	$\Uparrow$ 0.00	$\Uparrow$ 0.00
3	$\Rightarrow$ 0.00	$\Uparrow$ 0.00	$\Leftarrow$ 0.00	$\Leftarrow$ 0.00
2	$\Uparrow$ 0.00	$\Uparrow$ 0.00	$\Uparrow$ 0.00	$\Leftarrow$ 0.00
1	$\Rightarrow^{s_I}$ 0.00	$\Rightarrow$ 0.00	$\Uparrow$ 0.00	$\Leftarrow$ 0.00

$\gamma = 0.9$

$\epsilon = 0.001$

$\hat{V}_\pi^0$

- reward to `move` from striped cell is -3 (-1 everywhere else)
- `move` in gray cells unsuccessful with probability 0.6
- only applicable action in red cell is `noop` for reward of 0

Iterative Policy Evaluation: Example

5	⇒ −1.00	⇒ −1.00	⇒ −1.00	noop 0.00
4	⇒ −1.00	⇑ −1.00	⇑ −3.00	⇑ −1.00
3	⇒ −1.00	⇑ −1.00	⇐ −1.00	⇐ −1.00
2	⇑ −1.00	⇑ −1.00	⇑ −1.00	⇐ −1.00
1	⇒ s_I −1.00	⇒ −1.00	⇑ −1.00	⇐ −1.00
	1	2	3	4

$\gamma = 0.9$

$\epsilon = 0.001$

$\hat{V}_\pi^1$

max diff: 3.0

- reward to `move` from striped cell is -3 (-1 everywhere else)
- `move` in gray cells unsuccessful with probability 0.6
- only applicable action in red cell is `noop` for reward of 0

Iterative Policy Evaluation: Example

	1	2	3	4
5	$\Rightarrow$ −1.90	$\Rightarrow$ −1.90	$\Rightarrow$ −1.00	noop 0.00
4	$\Rightarrow$ −1.90	$\Uparrow$ −1.90	$\Uparrow$ −4.98	$\Uparrow$ −1.54
3	$\Rightarrow$ −1.90	$\Uparrow$ −1.90	$\Leftarrow$ −1.90	$\Leftarrow$ −1.90
2	$\Uparrow$ −1.90	$\Uparrow$ −1.90	$\Uparrow$ −1.90	$\Leftarrow$ −1.90
1	$\Rightarrow$ s_I −1.90	$\Rightarrow$ −1.90	$\Uparrow$ −1.90	$\Leftarrow$ −1.90

$\gamma = 0.9$

$\epsilon = 0.001$

$\hat{V}_\pi^2$

max diff: 1.98

- reward to `move` from striped cell is −3 (−1 everywhere else)
- `move` in gray cells unsuccessful with probability 0.6
- only applicable action in red cell is `noop` for reward of 0

Iterative Policy Evaluation: Example

	1	2	3	4
5	⇒ −3.38	⇒ −1.90	⇒ −1.00	noop 0.00
4	⇒ −3.83	⇑ −2.71	⇑ −6.94	⇑ −2.07
3	⇒ −4.10	⇑ −3.44	⇐ −4.10	⇐ −4.10
2	⇑ −4.10	⇑ −4.10	⇑ −4.10	⇐ −4.10
1	⇒ s_I −4.10	⇒ −4.10	⇑ −4.10	⇐ −4.10

$\gamma = 0.9$

$\epsilon = 0.001$

$\hat{V}_\pi^5$

max diff: 0.6561

- reward to move from striped cell is −3 (−1 everywhere else)
- move in gray cells unsuccessful with probability 0.6
- only applicable action in red cell is noop for reward of 0

Iterative Policy Evaluation: Example

	1	2	3	4
5	⇒ −3.65	⇒ −1.90	⇒ −1.00	noop 0.00
4	⇒ −4.27	⇑ −2.71	⇑ −7.29	⇑ −2.17
3	⇒ −4.83	⇑ −3.44	⇐ −4.10	⇐ −5.32
2	⇑ −5.78	⇑ −4.83	⇑ −4.69	⇐ −5.74
1	⇒ s_I −6.13	⇒ −5.70	⇑ −5.22	⇐ −6.09

$\gamma = 0.9$

$\epsilon = 0.001$

$\hat{V}_\pi^{10}$

max diff: 0.1395

- reward to `move` from striped cell is −3 (−1 everywhere else)
- `move` in gray cells unsuccessful with probability 0.6
- only applicable action in red cell is `noop` for reward of 0

Iterative Policy Evaluation: Example

	1	2	3	4
5	⇒ −3.66	⇒ −1.90	⇒ −1.00	`noop` 0.00
4	⇒ −4.29	⇑ −2.71	⇑ −7.30	⇑ −2.17
3	⇒ −4.87	⇑ −3.44	⇐ −4.10	⇐ −5.38
2	⇑ −5.98	⇑ −4.87	⇑ −4.69	⇐ −5.84
1	⇒ s_I −6.13	⇒ −5.70	⇑ −5.22	⇐ −6.26

$\gamma = 0.9$

$\epsilon = 0.001$

$\hat{V}_\pi^{20}$

max diff: 0.0007

- reward to `move` from striped cell is −3 (−1 everywhere else)
- `move` in gray cells unsuccessful with probability 0.6
- only applicable action in red cell is `noop` for reward of 0

Iterative Policy Evaluation: Properties

Theorem (Convergence of Iterative Policy Evaluation)

Let $\mathcal{T} = \langle S, A, R, T, s_I, \gamma \rangle$ be an MDP, π be a policy for $\mathcal{T}$ and $\hat{V}_\pi^0(s) \in \mathbb{R}$ arbitrarily for all $s \in S$.

Iterative policy evaluation converges to the true state-values, i.e.,

$$\lim_{i \to \infty} \hat{V}_\pi^i(s) = V_\pi(s) \text{ for all } s \in S.$$

In practice, iterative policy evaluation converges to true state-values if ϵ is small enough.

Policy Improvement

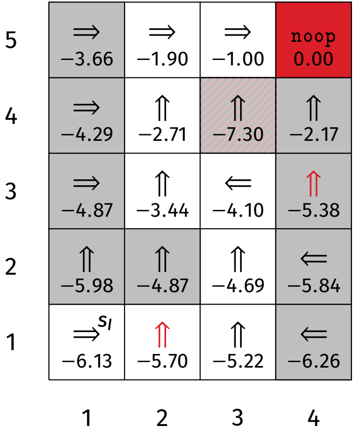
Greedy Action: Example

	1	2	3	4
5	$\Rightarrow$ −3.66	$\Rightarrow$ −1.90	$\Rightarrow$ −1.00	noop 0.00
4	$\Rightarrow$ −4.29	$\Uparrow$ −2.71	$\Uparrow$ −7.30	$\Uparrow$ −2.17
3	$\Rightarrow$ −4.87	$\Uparrow$ −3.44	$\Leftarrow$ −4.10	$\Leftarrow$ −5.38
2	$\Uparrow$ −5.98	$\Uparrow$ −4.87	$\Uparrow$ −4.69	$\Leftarrow$ −5.84
1	$\Rightarrow^{s_I}$ −6.13	$\Rightarrow$ −5.70	$\Uparrow$ −5.22	$\Leftarrow$ −6.26

$\gamma = 0.9$

- Can we learn more from this than the state-values of a policy?

Greedy Action: Example



$\gamma = 0.9$

- Can we learn more from this than the state-values of a policy?
- Yes! By evaluating all actions in each state, we can derive a better policy

Greedy Actions and Policies

Definition (Greedy Action)

Let s be a state of a MDP $\mathcal{T} = \langle S, A, R, T, s_I, \gamma \rangle$ and V be a state-value function for $\mathcal{T}$.

The set of greedy actions in s with respect to V is

$$A_V(s) := \arg\max_{a \in A(s)} \sum_{s' \in \mathrm{succ}(s,a)} T(s,a,s') \cdot (R(s,a,s') + \gamma \cdot V(s')) .$$

A policy π_V with $\pi_V(s) \in A_V(s)$ is a greedy policy.

Determining a greedy policy of a given state-value function is called policy improvement.

Policy Iteration

Policy Iteration

- policy iteration (PI) was first proposed by Howard in 1960
- based on the observation that greedy actions describe a better policy
- starts with arbitrary policy π_0
- alternates policy evaluation and policy improvement
- as long as policy changes

Example: Policy Iteration

	1	2	3	4
5	$\Rightarrow$ −3.66	$\Rightarrow$ −1.90	$\Rightarrow$ −1.00	noop 0.00
4	$\Rightarrow$ −4.29	$\Uparrow$ −2.71	$\Uparrow$ −7.30	$\Uparrow$ −2.17
3	$\Rightarrow$ −4.87	$\Uparrow$ −3.44	$\Leftarrow$ −4.10	$\Leftarrow$ −5.38
2	$\Uparrow$ −5.98	$\Uparrow$ −4.87	$\Uparrow$ −4.69	$\Leftarrow$ −5.84
1	$\Rightarrow$ s_I −6.13	$\Rightarrow$ −5.70	$\Uparrow$ −5.22	$\Leftarrow$ −6.26

$\gamma = 0.9$

π_0 and V_{π_0}

Example: Policy Iteration

5	$\Rightarrow$ −3.66	$\Rightarrow$ −1.90	$\Rightarrow$ −1.00	noop 0.00
4	$\Rightarrow$ −4.29	$\Uparrow$ −2.71	$\Uparrow$ −7.30	$\Uparrow$ −2.17
3	$\Rightarrow$ −4.87	$\Uparrow$ −3.44	$\Leftarrow$ −4.10	$\Uparrow$ −3.88
2	$\Uparrow$ −5.98	$\Uparrow$ −4.87	$\Uparrow$ −4.69	$\Leftarrow$ −5.84
1	$\Rightarrow^{s_I}$ −5.84	$\Uparrow$ −5.38	$\Uparrow$ −5.22	$\Leftarrow$ −6.26
	1	2	3	4

$\gamma = 0.9$

π_1 and V_{π_1}

Example: Policy Iteration

5	⇒ −3.66	⇒ −1.90	⇒ −1.00	noop 0.00
4	⇒ −4.29	⇑ −2.71	⇑ −7.30	⇑ −2.17
3	⇒ −4.87	⇑ −3.44	⇐ −4.10	⇑ −3.88
2	⇑ −5.98	⇑ −4.87	⇑ −4.69	⇑ −5.21
1	⇒ s_I −5.84	⇑ −5.38	⇑ −5.22	⇐ −6.26
	1	2	3	4

$\gamma = 0.9$

$\pi_2 = \pi_3$ and V_{π_2}

Computation of an Optimal Policy

there are various techniques to compute an optimal policy:

- ✓ (linear program)
- ✓ policy iteration
- value iteration

Value Iteration

From Policy Iteration to Value Iteration

- policy iteration:
  - search over policies
  - by evaluating their state-values
- value iteration:
  - search directly over state-values
  - optimal policy induced by final state-values

Value Iteration: Idea

- value iteration (VI) was first proposed by Bellman in 1957
- computes estimates $\hat{V}^0, \hat{V}^1, \ldots$ of $V_\star$ in an iterative process
- starts with arbitrary $\hat{V}^0$
- bases estimate $\hat{V}^{i+1}$ on values of estimate $\hat{V}^i$ by treating Bellman equation as update rule on all states:

$$\hat{V}^{i+1}(s) := \max_{a \in A(s)} \sum_{s' \in \text{succ}(s,a)} T(s,a,s') \cdot \left(R(s,a,s') + \gamma \hat{V}^i(s') \right)$$

- converges to state-values of optimal policy
- terminates when maximal change is smaller than residual ϵ

Value Iteration: Example

	1	2	3	4
5	0.00	0.00	0.00	noop 0.00
4	0.00	0.00	0.00	0.00
3	0.00	0.00	0.00	0.00
2	0.00	0.00	0.00	0.00
1	s_I 0.00	0.00	0.00	0.00

$\gamma = 0.9$

$\epsilon = 0.001$

$\hat{V}^0$

- reward to `move` from striped cell is -3 (-1 everywhere else)
- `move` in gray cells unsuccessful with probability 0.6
- only applicable action in red cell is `noop` for reward of 0

Value Iteration: Example

	1	2	3	4
5	−1.00	−1.00	−1.00	noop 0.00
4	−1.00	−1.00	−3.00	−1.00
3	−1.00	−1.00	−1.00	−1.00
2	−1.00	−1.00	−1.00	−1.00
1	s_I −1.00	−1.00	−1.00	−1.00

$\gamma = 0.9$

$\epsilon = 0.001$

$\hat{V}^1$

max diff: 3.0

- reward to `move` from striped cell is −3 (−1 everywhere else)
- `move` in gray cells unsuccessful with probability 0.6
- only applicable action in red cell is `noop` for reward of 0

Value Iteration: Example

5	−1.90	−1.90	−1.00	noop 0.00
4	−1.90	−1.90	−4.98	−1.54
3	−1.90	−1.90	−1.90	−1.90
2	−1.90	−1.90	−1.90	−1.90
1	s_I −1.90	−1.90	−1.90	−1.90
	1	2	3	4

$\gamma = 0.9$

$\epsilon = 0.001$

$\hat{V}^2$

max diff: 1.98

- reward to `move` from striped cell is −3 (−1 everywhere else)
- `move` in gray cells unsuccessful with probability 0.6
- only applicable action in red cell is `noop` for reward of 0

Value Iteration: Example

	1	2	3	4
5	−3.38	−1.90	−1.00	noop 0.00
4	−3.83	−2.71	−6.94	−2.07
3	−4.10	−3.44	−3.75	−3.36
2	−4.10	−4.10	−3.99	−3.93
1	s_I −4.10	−4.10	−4.10	−4.08

$\gamma = 0.9$

$\epsilon = 0.001$

$\hat{V}^5$

max diff: 0.6561

- reward to `move` from striped cell is −3 (−1 everywhere else)
- `move` in gray cells unsuccessful with probability 0.6
- only applicable action in red cell is `noop` for reward of 0

Value Iteration: Example

	1	2	3	4
5	−3.65	−1.90	−1.00	noop 0.00
4	−4.27	−2.71	−7.29	−2.17
3	−4.83	−3.44	−4.10	−3.84
2	−5.78	−4.83	−4.69	−5.05
1	s_I −5.74	−5.32	−5.22	−5.84

$\gamma = 0.9$

$\epsilon = 0.001$

$\hat{V}^{10}$

max diff: 0.187

- reward to `move` from striped cell is -3 (-1 everywhere else)
- `move` in gray cells unsuccessful with probability 0.6
- only applicable action in red cell is `noop` for reward of 0

Value Iteration: Example

	1	2	3	4
5	−3.66	−1.90	−1.00	noop 0.00
4	−4.29	−2.71	−7.30	−2.17
3	−4.87	−3.44	−4.10	−3.88
2	−5.98	−4.87	−4.69	−5.21
1	s_I −5.84	−5.38	−5.22	−6.25

$\gamma = 0.9$

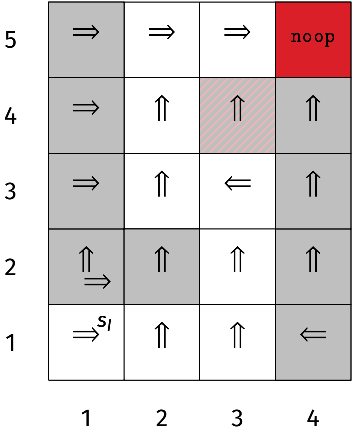
$\epsilon = 0.001$

$\hat{V}^{23}$

max diff: 0.0007

- reward to `move` from striped cell is −3 (−1 everywhere else)
- `move` in gray cells unsuccessful with probability 0.6
- only applicable action in red cell is `noop` for reward of 0

Value Iteration: Example

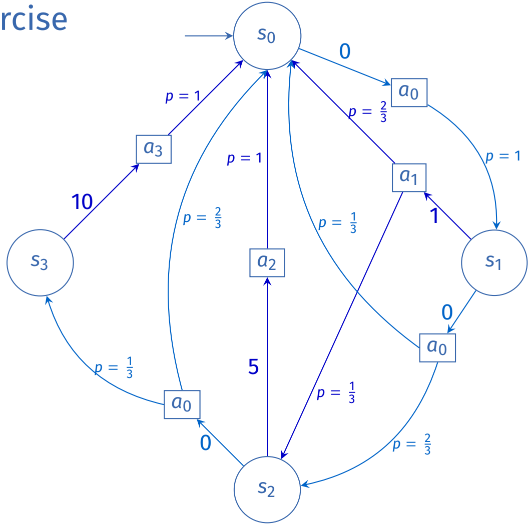


$\gamma = 0.9$

$\epsilon = 0.001$

- reward to `move` from striped cell is -3 (-1 everywhere else)
- `move` in gray cells unsuccessful with probability 0.6
- only applicable action in red cell is `noop` for reward of 0

Exercise



Using $\gamma = 0.9$ and the initial $\hat{V}^0$ values $\hat{V}^0(s_0) = 10$, $\hat{V}^0(s_1) = 0$, $\hat{V}^0(s_2) = 0$, $\hat{V}^0(s_3) = 0$, perform five iterations of value iteration.

Exercise: Solution

i	$\hat{V}^i(s_0)$	$\hat{V}^i(s_1)$	$\hat{V}^i(s_2)$	$\hat{V}^i(s_3)$
0	10	0	0	0
1	0	7	14	19
2	6.3	8.4	5.7	10
3	7.56	6.49	10.67	15.67
4	5.84	8.74	11.8	16.8
5	7.87	8.83	10.26	15.26

More about Planning

1. TDDD48 Automated Planning
2. MSc and PhD theses available in my group

Summary

Policy Iteration or Value Iteration?

- Policy evaluation is slightly cheaper than a VI iteration
  - PI faster than VI if few iterations required
- Asynchronous VI is basis of more sophisticated algorithm that can be applied in large MDPs and SSPs

Summary

- Policy iteration alternates policy evaluation and policy improvement.
- Value Iteration searches in the space of state-values and applies Bellman equation as update rule iteratively.

Quiz

Kahoot!