Artificial Intelligence

Planning: Solving MDPs

Jendrik Seipp

Linköping University

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

Introduction ●○○	Policy Evaluation	Policy Improvement	Policy Iteration	Value Iteration	Summ 0000
---------------------	-------------------	--------------------	------------------	-----------------	--------------

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 Improvement	Policy Iteration	
00000			

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:

- linear program
- ackward induction
- 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}_{π}^{0}
- treat state-value function as update rule

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

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

Policy Evaluation

Policy Improvement

Policy Iteration

alue Iteration

Summary 0000

Iterative Policy Evaluation: Example



$$\gamma = 0.9$$

 $\epsilon = 0.001$

 \hat{V}_{π}^{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

Policy Evaluation

Policy Improvement

Policy Iteration

alue Iteration

Summary 0000



- 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

Policy Evaluation

Policy Improvement

Policy Iteration

alue Iteration

Summary 0000



- 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

Policy Evaluation

Policy Improvement

Policy Iteration

alue Iteration

Summary 0000



- 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

Policy Evaluation

Policy Improvement

Policy Iteration

alue Iteration

Summary 0000



- 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

Policy Evaluation

Policy Improvement

Policy Iteration

alue Iteration

Summary 0000



- 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

/alue Iteration 000000 Summary 0000

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 $\hat{V}^0_{\pi}(s) \in \mathbb{R}$ arbitrarily for all $s \in S$.

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

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

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

Policy Evaluation	Policy Improvement	Policy Iteration	
	000 ·		

Summary 0000

Policy Improvement

Greedy Action: Example

5	⇒ -3.66	⇒ −1.90	⇒ −1.00	noop 0.00
4	⇒ −4.29		↑ -7.30	↑ -2.17
3	⇒ −4.87		← -4.10	← -5.38
2				← -5.84
1	⇒ ^{\$} ′ −6.13	⇒ -5.70		← -6.26
	1	2	3	4

 $\gamma = 0.9$

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

Greedy Action: Example

5	⇒ -3.66	⇒ −1.90	⇒ −1.00	noop 0.00
4	⇒ −4.29	↑ -2.71	↑ -7.30	↑ -2.17
3	⇒ −4.87		← -4.10	↑ -5.38
2				← -5.84
1	⇒ ^s ′ −6.13	↑ −5.70		← -6.26
	1	2	3	4

 $\gamma = 0.9$

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

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

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

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.

Introduction	Policy Evaluation	Policy Improvement	Policy Iteration ●○○○	Value Iteration 000000	Summary 0000

Introduction 000	Policy Evaluation	Policy Improvement	Policy Iteration ○●○○	Value Iteration 000000	Summa 0000

- 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

alue Iteration

Summary 0000

Example: Policy Iteration

5	⇒ −3.66	⇒ −1.90	⇒ −1.00	noop 0.00
4	⇒ -4.29		↑ -7.30	↑ -2.17
3	⇒ _4.87		← -4.10	← -5.38
2				← -5.84
1	⇒ ^{\$} ′ −6.13	⇒ -5.70		← -6.26
	1	2	3	4

 $\gamma = 0.9$

 π_0 and V_{π_0}

alue Iteration

Summary 0000

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		← -4.10	↑ -3.88
2				← -5.84
1	⇒ ^{\$} ′ −5.84	↑ −5.38		← -6.26
	1	2	3	4

 $\gamma = 0.9$

 π_1 and V_{π_1}

Example: Policy Iteration

5
$$\begin{array}{c|c} \Rightarrow & \Rightarrow \\ -3.66 & -1.90 & -1.00 & 0.00 \\ 4 & \Rightarrow & \uparrow \\ -4.29 & -2.71 & -7.30 & -2.17 \\ 3 & \Rightarrow & \uparrow \\ -4.87 & -3.44 & -4.10 & -3.88 \\ 2 & \uparrow \\ -5.98 & -4.87 & -4.69 & -5.21 \\ 1 & \Rightarrow^{S_{I}} & \uparrow \\ -5.84 & -5.38 & -5.22 & -6.26 \\ \end{array}$$

 $\gamma = 0.9$

$$\pi_2 = \pi_3 \text{ and } V_{\pi_2}$$

Introduction Policy Evaluation Policy Improvement Policy Iteration Value Iteration St 000 00000 000 0000 00000 00000000000	Value Iteration Sum	Policy Iteration 000●	Policy Improvement	Policy Evaluation	Introduction 000
---	---------------------	--------------------------	--------------------	-------------------	---------------------

Computation of an Optimal Policy

there are various techniques to compute an optimal policy:

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

Introduction 000	Policy Evaluation	Policy Improvement	Policy Iteration	Value Iteration •00000

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 \$\hit{\mathcal{V}}^{i+1}\$ on values of estimate \$\hit{\mathcal{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
- lacksquare terminates when maximal change is smaller than residual $m{\epsilon}$

	out	
oc	0	

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	
	SI				
1	0.00	0.00	0.00	0.00	
	1	2	3	4	

$$egin{array}{c} &= 0.9 \ arepsilon &= 0.001 \end{array}$$

 \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

5	-1.00	-1.00	-1.00	noop 0.00	$\gamma = 0.9$
4	-1.00	-1.00	-3.00	-1.00	e – 0.001
3	-1.00	-1.00	-1.00	-1.00	\hat{V}^1
2	-1.00	-1.00	-1.00	-1.00	may diffe 2.0
1	SI				max um: 5.0
I	-1.00	-1.00	-1.00	-1.00	
	1	2	3	4	

- 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

5	-1.90	-1.90	-1.00	noop 0.00	$\gamma = 0.9$
4	-1.90	-1.90	-4.98	-1.54	$\epsilon = 0.001$
3	-1.90	-1.90	-1.90	-1.90	\hat{V}^2
2	-1.90	-1.90	-1.90	-1.90	1.2.4.00
1	SI				max diff: 1.98
I	-1.90	-1.90	-1.90	-1.90	
	1	2	3	4	

- 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

5	-3.38	-1.90	-1.00	noop 0.00	$\gamma = 0.9$
4	-3.83	-2.71	-6.94	-2.07	6 - 0.001
3	-4.10	-3.44	-3.75	-3.36	\hat{V}^5
2	-4.10	-4.10	-3.99	-3.93	may diffe 0 (F(1
1	SI				max diff: 0.6561
I	-4.10	-4.10	-4.10	-4.08	
	1	2	3	4	

- 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

5	-3.65	-1.90	-1.00	noop 0.00	$\gamma = 0.9$
4	-4.27	-2.71	-7.29	-2.17	$\epsilon = 0.001$
3	-4.83	-3.44	-4.10	-3.84	Ŷ ¹⁰
2	-5.78	-4.83	-4.69	-5.05	
1	SI				max diff: 0.187
I	-5.74	-5.32	-5.22	-5.84	
	1	2	3	4	

- 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

5	-3.66	-1.90	-1.00	noop 0.00	$\gamma = 0.9$
4	-4.29	-2.71	-7.30	-2.17	8 – 0.001
3	-4.87	-3.44	-4.10	-3.88	\hat{V}^{23}
2	-5.98	-4.87	-4.69	-5.21	may diff. 0 0007
1	SI				max diii: 0.0007
I	-5.84	-5.38	-5.22	-6.25	
	1	2	3	4	

- 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



$$\gamma = 0.9$$

 $\epsilon = 0.001$

- 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



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.

Introduction 000	Policy Evaluation	Policy Improvement	Policy Iteration	Value Iteration ○○○○○●	Summary 0000

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

Introduction 000	Policy Evaluation	Policy Improvement	Policy Iteration	Value Iteration	Summary 0000

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

ntroduction	Policy Evaluation	Policy Improvement	Policy Iteration	Value Iteratio

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.

Summary

More about AI Planning

- TDDD48 Automated Planning (Spring semester)
- MSc and PhD theses in my group: https://mrlab.ai/positions