Поиск оптимальных стратегий в МППР

А.Г. Трофимов
к.т.н., доцент, НИЯУ МИФИ

lab@neuroinfo.ru
http://datalearning.ru

Курс “Статистическая обработка временных рядов”

Сентябрь 2018

Policy for MDP

Markov Reward Process (MRP) is a Markov chain (discrete-time or continuous-time) with an associated reward to each state

Markov Decision Process (MDP) is a Markov reward process with actions. Actions are taken by a decision maker

If the decision maker is following policy π at time step $n \in \{0, 1, ...\}$, then

$$\pi(a|s) = P(A_n = a \mid X_n = s), \quad a \in A(s),\ s \in S$$

As soon as a policy π is applied to MDP, it becomes an MRP:

$$\text{MDP} + \text{policy} = \text{MRP}$$

The core problem of MDPs is to find a policy π for the decision maker

Better Policy

Given two policies π and π' which one is better?

The decision maker is interested in finding a policy that achieves a lot of reward over the long run

Definition

A policy π is better than a policy π' if value function $v_\pi(s)$ is greater than or equal to value function $v_{\pi'}(s)$ for all states $s \in S$:

$$\pi \geq \pi' \Leftrightarrow v_\pi(s) \geq v_{\pi'}(s) \quad \forall s \in S$$

To be a better policy, it can't be worse than another policy in any state $s \in S$

A policy has to get at least as much reward as another policy in all states to say it's an equally good policy, and more reward in at least one state to say it's a better policy

Howard's Theorem

Definition

A policy π_* is optimal policy if it is better or equal to all other policies π: $\pi_* \geq \pi, \quad \forall \pi$

Theorem (Howard, 1960)

For any finite MDP with finite reward function $\rho(s, a)$:

- At least one optimal policy π_* exists
- All optimal policies are deterministic (if there are no limitations on policy space)
- All optimal policies share the same state-value function $v_*(s)$, called the optimal state-value function
- All optimal policies share the same action-value function $q_*(s, a)$, called the optimal action-value function

Optimal State-Value and Action-Value Functions

The optimal policy π_* gives the optimal expected rewards in long-term run

The optimal state-value function:

$$v_*(s) = \max_\pi v_\pi(s) \quad \forall s \in S$$

It gives the expected return starting from state s and thereafter following an optimal policy π_*

The optimal action-value function:

$$q_*(s,a) = \max_\pi q_\pi(s,a) \quad \forall s \in S,\ a \in A(s)$$

It gives the expected return for taking action a in state s and thereafter following an optimal policy π_*

Optimal Policy

The optimal policy:

$$\pi_* = \arg\max_{\pi} v_\pi(s) = \arg\max_{\pi} q_\pi(s, a)$$

(by Howard's theorem)

Therefore, the optimal policy π_* is the policy that takes the best action $a \in A(s)$ in every state $s \in S$:

$$\pi_*(a|s) = \begin{cases} 1, & a = \arg\max\limits_{a \in A(s)} q_*(s, a), \\ 0, & \text{otherwise} \end{cases}$$

As soon as optimal action-value function $q_*(s, a)$ is known, we know which action $a \in A(s)$ to choose in every state $s \in S$ to follow the optimal policy π_*

Therefore, the optimal policy π_* should be a deterministic policy

How to find the optimal action-value function $q_*(s, a)$?

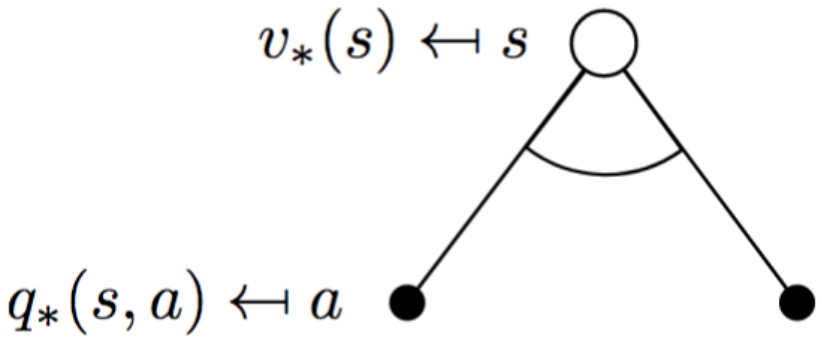
Optimal State-Value Function

State-value function for MDP with policy π:

$$v_\pi(s_i) = \mathrm{M}_\pi[G_n \mid X_n = s_i] = \sum_{a \in A(s_i)} q_\pi(s_i, a)\pi(a|s_i), \quad i = 1, ..., k$$

Under deterministic optimal policy π_* we know which action to choose in every state $s_i \in S$; the expectation over the values of next successor actions is reduced to the $\max$ operation:

$$v_*(s_i) = \max_{a \in A(s_i)} q_*(s_i, a)$$



The arc between action nodes represents the $\max$ function

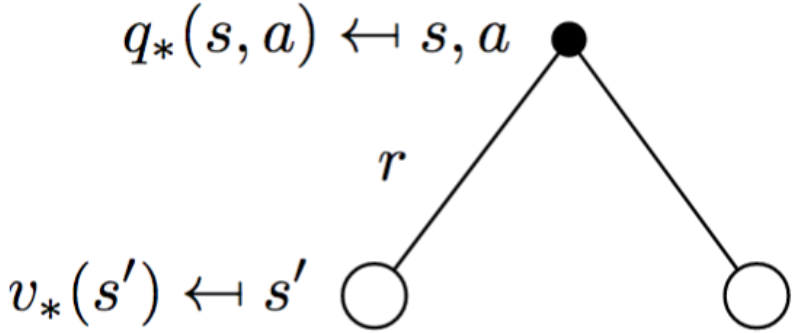
Optimal Action-Value Function

Action-value function for MDP with policy π:

$$q_\pi(s_i, a) = \mathrm{M}_\pi[G_n \mid X_n = s_i, A_n = a] = \rho(s_i, a) + \gamma \sum_{j=1}^{k} p_{a,ij} v_\pi(s_j)$$

The successor state s_j due to action a is defined by state transition probabilities $p_{a,ij}$, $j = 1, ..., k$, it doesn't depend on the policy; for optimal policy π_* the expectation is still needed:

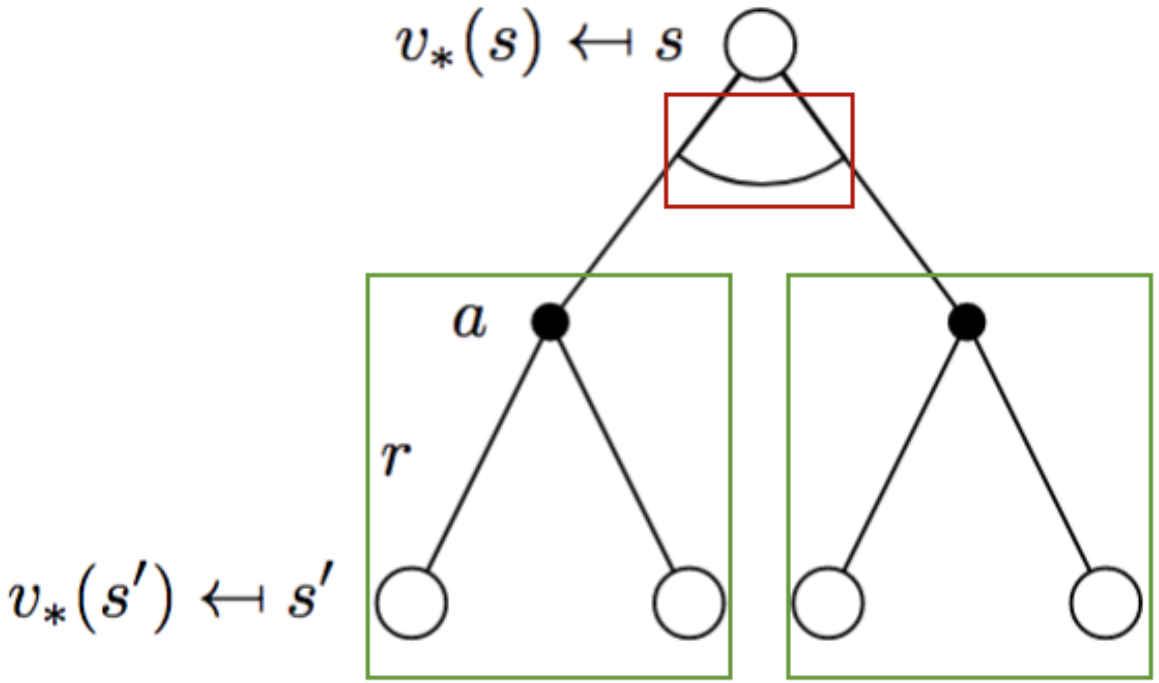
$$q_*(s_i, a) = \mathrm{M}_{\pi_*}[G_n \mid X_n = s_i, A_n = a] = \rho(s_i, a) + \gamma \sum_{j=1}^{k} p_{a,ij} v_*(s_j)$$



Bellman Optimality Equation for State-Value Function

Recurrent expression for optimal state-value function:

$$v_*(s_i) = \max_{a \in A(s_i)} q_*(s_i, a) = \max_{a \in A(s_i)} \left(\rho(s_i, a) + \gamma \sum_{j=1}^{k} p_{a,ij} v_*(s_j) \right)$$

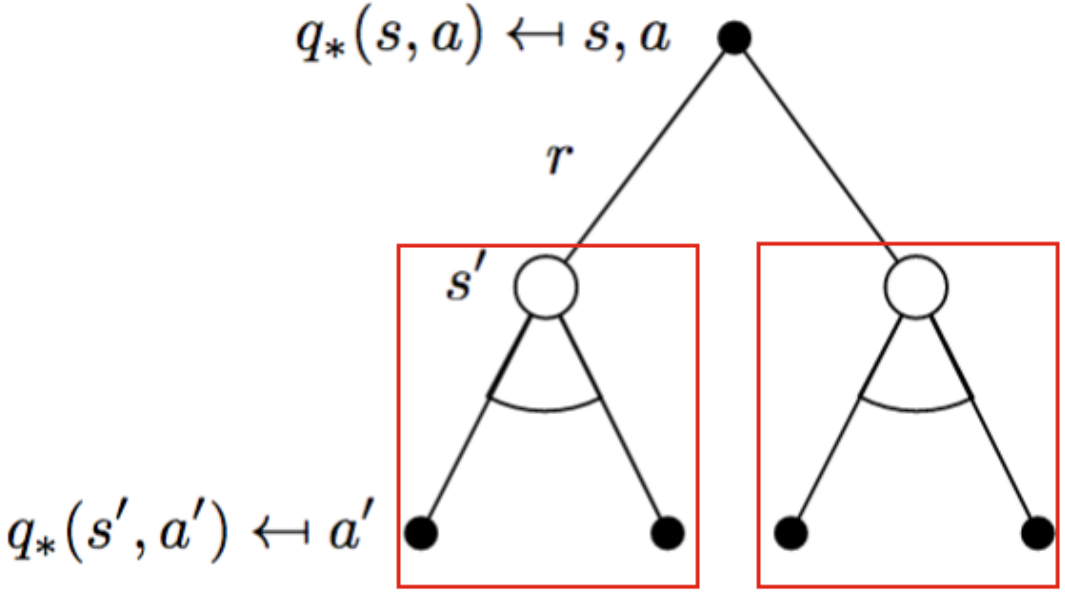


Bellman Optimality Equation for Action-Value Function

Recurrent expression for optimal action-value function:

$$q_*(s_i, a) = \rho(s_i, a) + \gamma \sum_{j=1}^{k} p_{a,ij} v_*(s_j)$$

$$= \rho(s_i, a) + \gamma \sum_{j=1}^{k} p_{a,ij} \max_{a' \in A(s_j)} q_*(s_j, a')$$



Bellman Optimality Equations

Bellman optimality equation for state-value function:

$$v_*(s_i) = \max_{a \in A(s_i)} \left(\rho(s_i, a) + \gamma \sum_{j=1}^{k} p_{a,ij} v_*(s_j) \right), \quad i = 1, ..., k$$

Bellman optimality equation for action-value function:

$$q_*(s_i, a) = \rho(s_i, a) + \gamma \sum_{j=1}^{k} p_{a,ij} \max_{a' \in A(s_i)} q_*(s_i, a'), \quad i = 1, ..., k$$

Bellman optimality equations are non-linear equations due to $\max$ operation, there are no closed-form solutions

Iterative algorithms are used to solve them

Greedy Policy

The term greedy is used to describe any decision procedure that selects alternatives based only on local or immediate considerations, without considering the possibility that such a selection may prevent future access to even better alternatives

Greedy policy for MDP:
Take action $a \in A(s)$ in every state $s \in S$ that leads to the best reward function $\rho(s, a)$:

$$\pi(a|s) = \begin{cases} 1, & a = \arg \max\limits_{a \in A(s)} \rho(s, a), \\ 0, & \text{otherwise} \end{cases}$$

This policy maximizes immediate reward at each step, it is "myopic" policy. There is no guarantee that it also maximizes long-term cumulative reward

Optimal Policy vs Greedy Policy

Optimal policy for MDP:
Take action $a \in A(s)$ in every state $s \in S$ that leads to the best optimal action-value function $q_*(s, a)$:

$$\pi_*(a|s) = \begin{cases} 1, & a = \arg\max\limits_{a \in A(s)} q_*(s, a), \\ 0, & \text{otherwise} \end{cases}$$

This policy maximizes future cumulative reward at each step, it is "far-sighted" policy

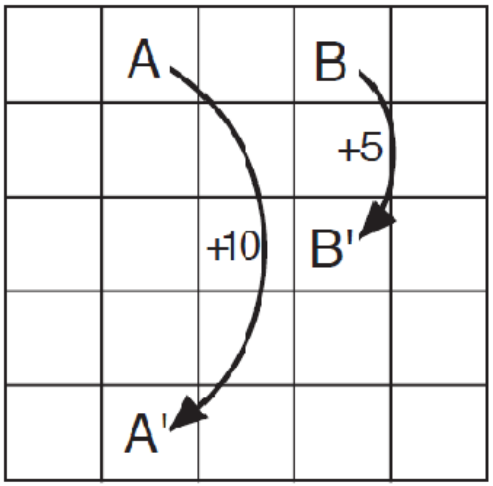
If optimal action-value function $q_*(s, a)$ is known, than optimal policy π_* is equivalent to the greedy policy for MDP with reward function $\rho(s, a) \equiv q_*(s, a)$

It's important to solve Bellman optimality equation and find optimal action-value function $q_*(s, a)$!

Gridworld Example. Random Policy

Suppose the decision maker applies the policy π_0:
randomly and equally probable choose an action from set $A(s) = \{\text{north, south, east, west}\}$ in all states $s \in S$

Reward function $\rho_0(s)$ and state-value function $v_0(s)$ for random policy π_0 for gridworld MDP with $\gamma = 0.9$



−0.5	10.0	−.25	5.0	−0.5
−.25	0	0	0	−.25
−.25	0	0	0	−.25
−.25	0	0	0	−.25
−0.5	−.25	−.25	−.25	−0.5

ρ_0

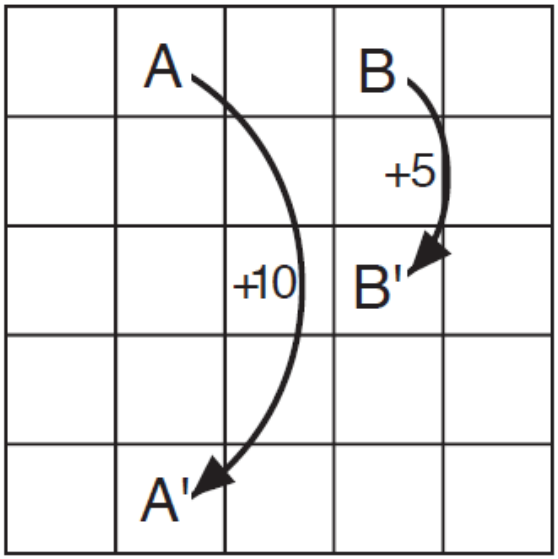
3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

v_0

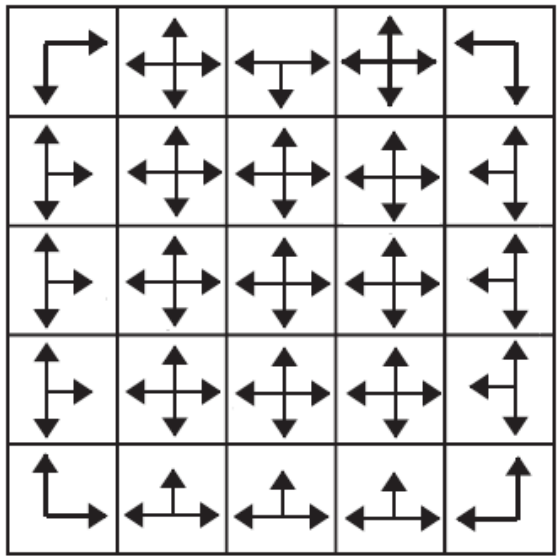
The maximal expected return +8.8 can be achieved from state A

Gridworld Example. Greedy Policy 1

Greedy policy π_1 w.r.t. immediate rewards $\rho(s, a)$



Gridworld

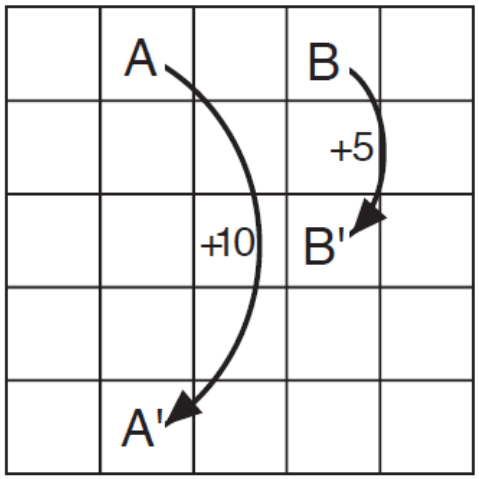


π_1

The greedy policy π_1 takes action $a = \pi_1(s)$ in each state $s \in S$ that leads to receive highest immediate reward (i.e. reward function $\rho(s, a)$)

Gridworld Example. Greedy Policy 2

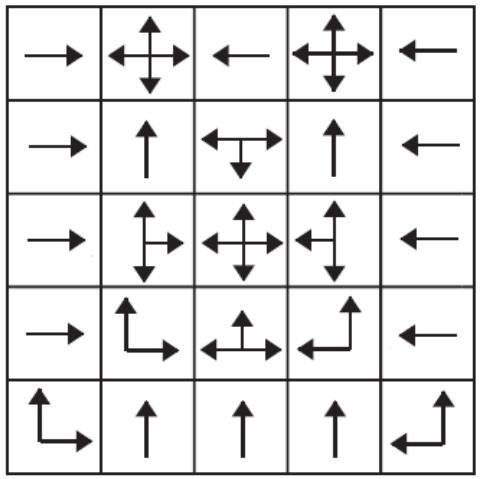
Greedy policy π_2 w.r.t. reward function $\rho_0(s)$



Gridworld

−0.5	10.0	−.25	5.0	−0.5
−.25	0	0	0	−.25
−.25	0	0	0	−.25
−.25	0	0	0	−.25
−0.5	−.25	−.25	−.25	−0.5

ρ_0

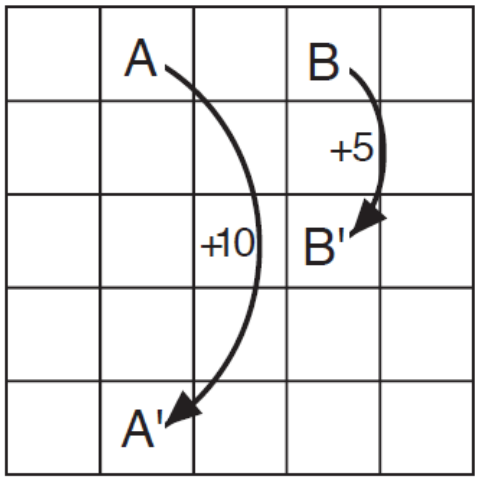


π_2

The greedy policy π_2 takes action $\pi_2(s)$ in each state $s \in S$ that jumps the MDP to the state with highest immediate expected reward w.r.t. random policy π_0 (i.e. reward function $\rho_0(s)$)

Gridworld Example. Optimal Policy

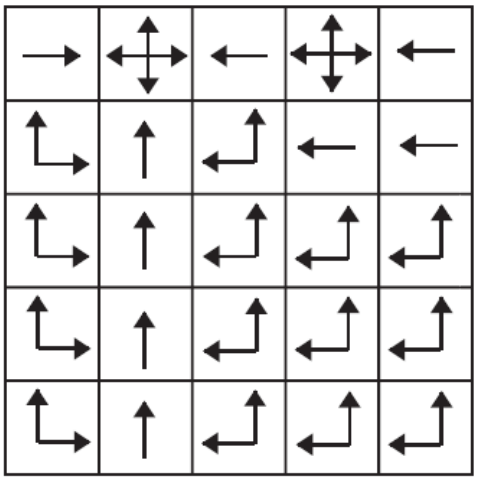
Optimal state-value function $v_*(s)$ and optimal policy π_* for gridworld MDP with $\gamma = 0.9$



Gridworld

22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

v_*



π_*

The optimal policy π_* takes action $\pi_*(s)$ in each state $s \in S$ that jumps the MDP to the state with highest long-term expected reward w.r.t. policy π_* (i.e. state-value function $v_*(s)$)

Gridworld Example. Notes

- Where there are multiple arrows in a cell, all of the corresponding actions are optimal
- The values $v_*(s)$ (i.e. expected discounted return) under optimal policy π_* are highest for all states $s \in S$ among any other policies π
- The stationary behaviour of MDP with optimal policy π_* is a cycle:

$$A - A' - \text{up} - \text{up} - \text{up} - \text{up} - A$$

- The value of state A w.r.t. policy π_*:

$$v_*(A) = 10(1 + 0.9^5 + 0.9^{10} + ...) \approx 24.4$$

Bellman Equation for MDP with Policy

The Bellman equation for MDP with policy π:

$$v_\pi(s_i) = \rho_\pi(s_i) + \gamma \sum_{j=1}^{k} p_{\pi,ij} v_\pi(s_j), \quad i = 1, ..., k$$

In matrix form:

$$v_\pi = \rho_\pi + \gamma P_\pi v_\pi$$

It is a linear equation and it has a closed form solution:

$$v_\pi = (I - \gamma P_\pi)^{-1} \rho_\pi$$

In practice, it can be solved only for small number of states k, because of the inverse operation (complexity $O(k^3)$)

Iterative algorithms are used

Iterative Policy Evaluation

The Bellman equation for MDP with policy π:

$$v_\pi = \rho_\pi + \gamma P_\pi v_\pi$$

Iterative algorithm of solving:

- **Step 1.** Initialize $v_0 = 0$
- **Step 2.** Compute v_n at current iteration n:

$$v_n = \rho_\pi + \gamma P_\pi v_{n-1}$$

- **Step 3.** Repeat step 2 until convergence

This algorithm is called iterative policy evaluation

It can be shown that the sequence $v_0, v_1, ...$ converges to v_π as $n \to \infty$:

$$\lim_{n\to\infty} v_n = v_\pi$$

If $v_n = v_\pi$, than $v_{n+1} = v_{n+2} = ... = v_\pi$

Iterative Policy Evaluation. Notes

- Iterative policy evaluation is a form of fixed point iteration method used to find the fixed point of a function:

$$v_\pi = f(v_\pi), \quad \text{where} \quad f(v) = \rho_\pi + \gamma P_\pi v$$

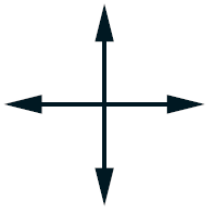
- Can be used different stopping criteria for iterative policy evaluation, e.g.:

$$\max_{s \in S} |v_n(s) - v_{n-1}(s)| < \Delta$$

  where Δ is a small positive constant
- In-place version of iterative policy evaluation:
  at each iteration the new value $v_n(s_i)$ immediately overwrites the old one and is used to calculate $v_n(s_j)$ at the same iteration
- In-place algorithm also converges to v_π, usually faster, since it uses new data as soon as they are available

Gridworld Example. Description

Consider an MDP with states that correspond to the cells of a gridworld. At each cell four actions are possible: north, south, east and west, which deterministically cause the chain to move one cell in the respective direction on the grid



actions

	1	2	3
4	5	6	7
8	9	10	11
12	13	14	

$R_t = -1$
on all transitions

$k = |S| = 14 + 1 = 15$,
$A(s) = \{\text{north, south, east, west}\}$ for all $s \in S$
All transition probabilities $p_{a,ij}$ are equal to 1 or 0 for all $s_i \in S$, $s_j \in S$, $a \in A(s_i)$, $i, j = 0, ..., 14$

Gridworld Example. Rewards and Policy

The reward is –1 on all transitions until the terminal state s_0 (shown as cells (1,1) and (4,4)) is reached:

$$R(s_i, a, s_j) = -1$$

for all states s_i, s_j and actions $a \in A(s_i)$ except the terminal state $s_i = s_0$ with zero reward

Suppose the decision maker applies the random policy π: randomly and equally probable choose an action from set $A(s) = \{\text{north, south, east, west}\}$ in all states $s \in S$

Assume discount rate $\gamma = 1$ (no discounting of reward)

The cumulative penalty (negative reward) of an episode is a number of steps passed to terminal state s_0

Gridworld Example. Reward Function

The reward function for MDP with equiprobable random policy π:

$$\rho_\pi(s_i) = -1$$

for all states $s_i \in S$ except the terminal state s_0

In matrix form: $\rho_\pi(s) = (0, -1, ..., -1)^T$

Reward function

$\rho_\pi(s)$

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0

Exact state-value function

$v_\pi(s)$

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0

Iterative policy evaluation formula:

$$v_n = \rho_\pi + \gamma P_\pi v_{n-1}$$

Gridworld Example. Iterative Policy Evaluation

$k = 0$

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0

$k = 1$

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0

$k = 2$

0.0	-1.7	-2.0	-2.0
-1.7	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.7
-2.0	-2.0	-1.7	0.0

$k = 3$

0.0	-2.4	-2.9	-3.0
-2.4	-2.9	-3.0	-2.9
-2.9	-3.0	-2.9	-2.4
-3.0	-2.9	-2.4	0.0

$k = 10$

0.0	-6.1	-8.4	-9.0
-6.1	-7.7	-8.4	-8.4
-8.4	-8.4	-7.7	-6.1
-9.0	-8.4	-6.1	0.0

$k = \infty$

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0

Policy Improvement Theorem

Should we change the action $a = \pi(s)$ in s to some other action $a' \neq \pi(s)$ and select it every time the state s is encountered? Would it be better or worse to change policy π to new policy π' that differs from π in an action at one state $s \in S$?

Policy Improvement Theorem

Let π and π' be any pair of deterministic policies such that

$$q_\pi(s, \pi'(s)) \geq v_\pi(s) \quad \text{for all } s \in S$$

where $\pi'(s)$ is action to be taken in state s under policy π'. Then the policy π' must be as good as, or better than, π. That is, it must obtain greater or equal expected return from all states $s \in S$:

$$v_{\pi'}(s) \geq v_\pi(s)$$

Policy Improvement Theorem. Notes

- The condition $q_\pi(s, \pi'(s)) \geq v_\pi(s)$ for all $s \in S$ means that the expected return received by taking action $\pi'(s)$ in s once and thereafter following the existing policy π is better than always to follow policy π
- The policy improvement theorem shows that it is better to select $\pi'(s)$ in s every time the state s is encountered, i.e. to change policy π to π'
- If $q_\pi(s, \pi'(s)) > v_\pi(s)$ at least at one state $s \in S$, then policy π' is better than π, i.e. $v_{\pi'}(s) > v_\pi(s)$ at least at one state $s \in S$
- The policy improvement theorem can be generalized to stochastic policies π and π'

Policy Improvement

Given a policy π and its value function $v_\pi(s)$, we can easily improve it by changing the action $\pi(s_i)$ at each state $s_i \in S$ to the action $\pi'(s_i)$ that appears best according to $q_\pi(s_i, a)$:

$$\pi'(s_i) = \arg \max_{a \in A(s_i)} q_\pi(s_i, a) = \arg \max_{a \in A(s_i)} \left(\rho(s_i, a) + \gamma \sum_{j=1}^{k} p_{a,ij} v_\pi(s_j) \right)$$

This process is called policy improvement

The policy π' takes the action in $s \in S$ that looks best in the short term, after one step of lookahead, according to v_π

The policy improvement improves an original policy in a greedy manner w.r.t. the value function v_π of the original policy π

Fixed Point of Policy Improvement

Suppose the new policy π' is as good as, but not better than, the old policy π:

$$v_{\pi'} = v_{\pi}$$

Policy improvement:

$$\begin{aligned}
\pi'(s_i) &= \arg \max_{a \in A(s_i)} q_{\pi}(s_i, a) \\
&= \arg \max_{a \in A(s_i)} \left(\rho(s_i, a) + \gamma \sum_{j=1}^{k} p_{a,ij} v_{\pi}(s_j) \right) \\
&= \arg \max_{a \in A(s_i)} \left(\rho(s_i, a) + \gamma \sum_{j=1}^{k} p_{a,ij} v_{\pi'}(s_j) \right) \qquad \text{for all } s_i \in S
\end{aligned}$$

Policy Improvement and Bellman Optimality Equation

Bellman optimality equation for state-value function:

$$v_*(s_i) = \max_{a \in A(s_i)} \left(\rho(s_i, a) + \gamma \sum_{j=1}^{k} p_{a,ij} v_*(s_j) \right), \quad s_i \in S$$

$$\pi_*(s_i) = \arg \max_{a \in A(s_i)} \left(\rho(s_i, a) + \gamma \sum_{j=1}^{k} p_{a,ij} v_*(s_j) \right), \quad s_i \in S$$

Therefore, π' must be π_* and both π and π' must be optimal policies

Policy improvement thus must give us a strictly better policy except when the original policy is already optimal

Policy Iteration

Once a policy π has been improved using v_π to yield a better policy π', we can then compute $v_{\pi'}$ and improve it again to yield an even better π''. Thus a sequence of monotonically improving policies and value functions will be obtained:

$$\pi_0 \xrightarrow{E} v_{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} v_{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} \ldots \xrightarrow{I} \pi_* \xrightarrow{E} v_*$$

where $\xrightarrow{E}$ denotes a policy evaluation and $\xrightarrow{I}$ denotes a policy improvement

Each policy is guaranteed to be a strict improvement over the previous one (unless it is already optimal)

Because a finite MDP has only a finite number of policies, this process must converge to an optimal policy π_* and optimal value function $v_*(s)$ in a finite number of iterations

This way of finding an optimal policy is called policy iteration

Policy Iteration Algorithm

- **Step 1.** Initialization
  Initialize $v_0(s) \in \mathbb{R}$ and $\pi_0(s) \in A(s)$ randomly for each $s \in S$
- **Step 2.** Policy evaluation
  Perform iterative policy evaluation from the initial value function v_{n-1} for the previous policy:

$$\pi_{n-1}, v_{n-1} \xrightarrow{E} v_n$$

- **Step 3.** Policy improvement
  Improve current policy π_n in a greedy manner w.r.t. value function v_n:

$$v_n \xrightarrow{I} \pi_n$$

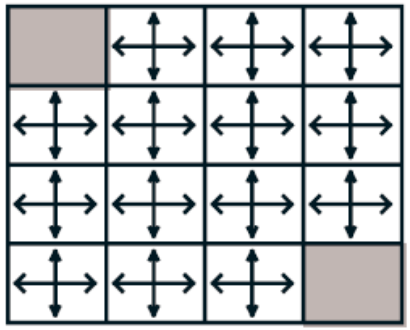
- **Step 4.** Repeat steps 2,3 until convergence

Gridworld Example. Policy Iteration

1. Initialization

$k = 0$

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0



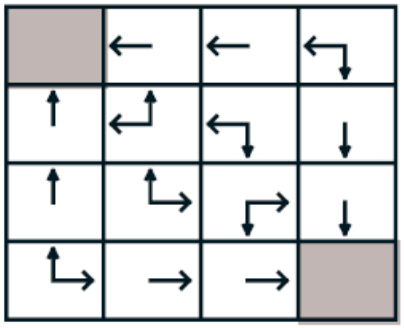
random policy

2. Policy evaluation

$k = \infty$

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0

3. Policy improvement



At next iteration the state-value function remains unchanged, therefore, the policy after first improvement is optimal

Policy Iteration. Notes

- Policy iteration often converges in surprisingly few iterations For gridworld example the optimal policy was founded after just one iteration
- At each iteration the policy evaluation procedure starts with the value function for the previous policy. It typically results in a great increase in the speed of convergence of policy evaluation (presumably because the value function changes little from one policy to the next)
- A major drawback of policy iteration is that each of its iterations involves policy evaluation, which is iterative procedure itself
- The convergence of policy iteration is guaranteed only in the limit. Must we wait for exact convergence, or can we stop earlier?

Value Iteration

It can be shown that the policy evaluation step of policy iteration can be truncated in several ways without losing the convergence guarantees of policy iteration

Particularly, the policy evaluation can be stopped after first iteration

This way of finding an optimal policy is called value iteration

Notes:

- Value iteration algorithm is policy iteration with truncated policy evaluation step to just one policy update
- Faster convergence is often achieved by interposing multiple policy evaluation updates between each policy improvement update
- All of these algorithms with arbitrary number of updates at policy evaluation step converge to an optimal policy for discounted ($0 \leq \gamma < 1$) finite MDPs

Value Iteration Formula

Truncated policy evaluation:

$$v_n = \rho_{\pi_{n-1}} + \gamma P_{\pi_{n-1}} v_{n-1}$$

$$v_n(s_i) = \rho_{\pi_{n-1}}(s_i) + \gamma \sum_{j=1}^{k} p_{\pi_{n-1},ij} v_{n-1}(s_j), \quad i = 1, ..., k$$

Policy improvement:

$$\pi_n(s_i) = \arg \max_{a \in A(s_i)} \left(\rho(s_i, a) + \gamma \sum_{j=1}^{k} p_{a,ij} v_n(s_j) \right), \quad s_i \in S$$

Value iteration:

$$v_{n+1}(s_i) = \max_{a \in A(s_i)} \left(\rho(s_i, a) + \gamma \sum_{j=1}^{k} p_{a,ij} v_n(s_j) \right), \quad s_i \in S$$

Value Iteration Algorithm

- **Step 1.** Initialize $v_0 = 0$
- **Step 2.** Compute v_n at current iteration n:

$$v_n(s_i) = \max_{a \in A(s_i)} \left(\rho(s_i, a) + \gamma \sum_{j=1}^{k} p_{a,ij} v_{n-1}(s_j) \right), \quad s_i \in S$$

- **Step 3.** Repeat step 2 until convergence
- **Step 4.** Output a deterministic policy

$$\pi(s_i) = \arg \max_{a \in A(s_i)} v_n(s_i)$$

$$= \arg \max_{a \in A(s_i)} \left(\rho(s_i, a) + \gamma \sum_{j=1}^{k} p_{a,ij} v_{n-1}(s_j) \right), \quad s_i \in S$$

Value Iteration and Bellman Optimality Equation

Value iteration:

$$v_n(s_i) = \max_{a \in A(s_i)} \left(\rho(s_i, a) + \gamma \sum_{j=1}^{k} p_{a,ij} v_{n-1}(s_j) \right), \quad s_i \in S$$

Bellman optimality equation for state-value function:

$$v_*(s_i) = \max_{a \in A(s_i)} \left(\rho(s_i, a) + \gamma \sum_{j=1}^{k} p_{a,ij} v_*(s_j) \right), \quad s_i \in S$$

Value iteration is obtained simply by turning the Bellman optimality equation into an update rule

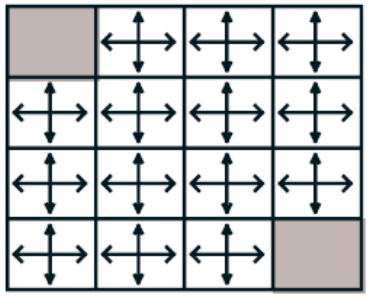
It's guaranteed that value iteration will find an optimal policy v_* in the limit. In practice, the procedure is stopped by criterion:

$$\max_{s \in S} |v_n(s) - v_{n-1}(s)| < \Delta, \quad \text{where } \Delta \text{ is a small contant}$$

Gridworld Example. Value Iteration

$k = 0$

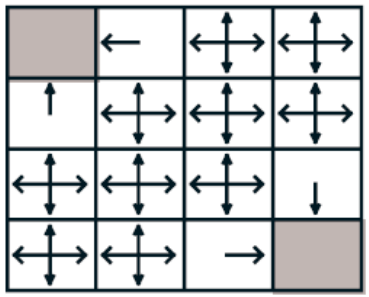
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0



random policy

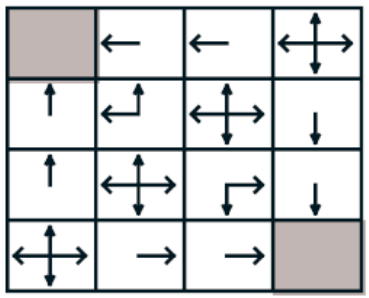
$k = 1$

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0



$k = 2$

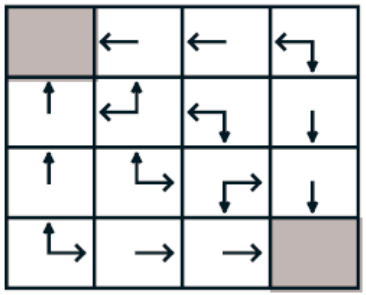
0.0	-1.0	-2.0	-2.0
-1.0	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.0
-2.0	-2.0	-1.0	0.0



Gridworld Example. Value Iteration

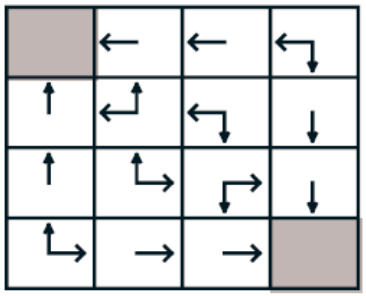
$k = 3$

0.0	-1.0	-2.0	-3.0
-1.0	-2.0	-3.0	-2.0
-2.0	-3.0	-2.0	-1.0
-3.0	-2.0	-1.0	0.0



$k = 10$

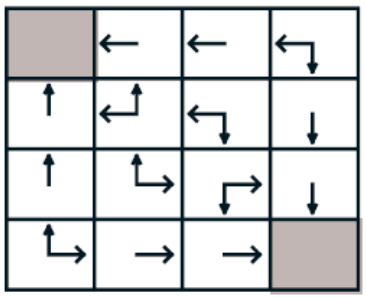
0.0	-1.0	-2.0	-3.0
-1.0	-2.0	-3.0	-2.0
-2.0	-3.0	-2.0	-1.0
-3.0	-2.0	-1.0	0.0



optimal policy

$k = \infty$

0.0	-1.0	-2.0	-3.0
-1.0	-2.0	-3.0	-2.0
-2.0	-3.0	-2.0	-1.0
-3.0	-2.0	-1.0	0.0



Asynchronous Value Iteration

A major drawback of value iteration is that it involves operations over the entire state set of the MDP

If the state set is very large, then even a single update can be prohibitively expensive (e.g., the game of backgammon has over 10^{20} states)

In-place version of value iteration:
The state-value function $v(s)$ can be updated after computing for each state $s \in S$ or a some mini-batch of states

This algorithm of value iteration is called asynchronous value iteration

It's guaranteed asymptotic convergence to optimal state-value function v_* for discounted $(0 \leq \gamma < 1)$ finite MDP and stochastic sequence of states in asynchronous value iteration

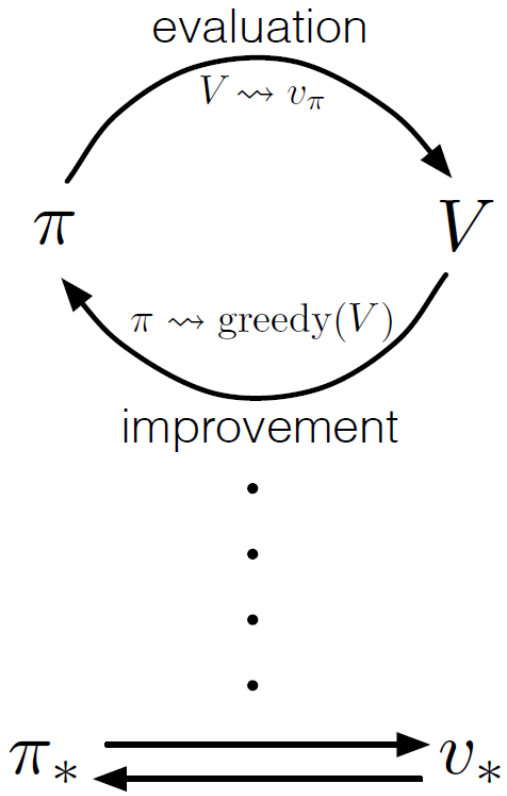
Asynchronous Value Iteration. Notes

- The values of some states may be updated several times before the values of others are updated once
- Some states may not need their values updated as often as others. We might even try to skip updating some states entirely if they are not relevant to optimal behaviour
- Asynchronous algorithms make it easier to intermix computation with real-time interaction. We can observe the optimization progress immediately
- Time required to one iteration of asynchronous optimization algorithm doesn't depend on the number of the MDP's states

Evaluation and Improvement Processes

Policy iteration consists of two simultaneous, interacting processes:

- Policy evaluation
  It makes the value function consistent with the current policy
- Policy improvement
  It makes the policy greedy with respect to the current value function



These processes continue up to convergence to optimal state-value function $v_*(s)$, $s \in S$, and optimal policy π_*

Policy Evaluation vs Policy Improvement

The evaluation and improvement processes in policy iteration are both competing and cooperating processes

Making the policy greedy with respect to the value function typically makes the value function incorrect for the changed policy, and making the value function consistent with the policy typically causes that policy no longer to be greedy, and the process repeats again until a joint solution is found: $v_*(s)$ and π_*