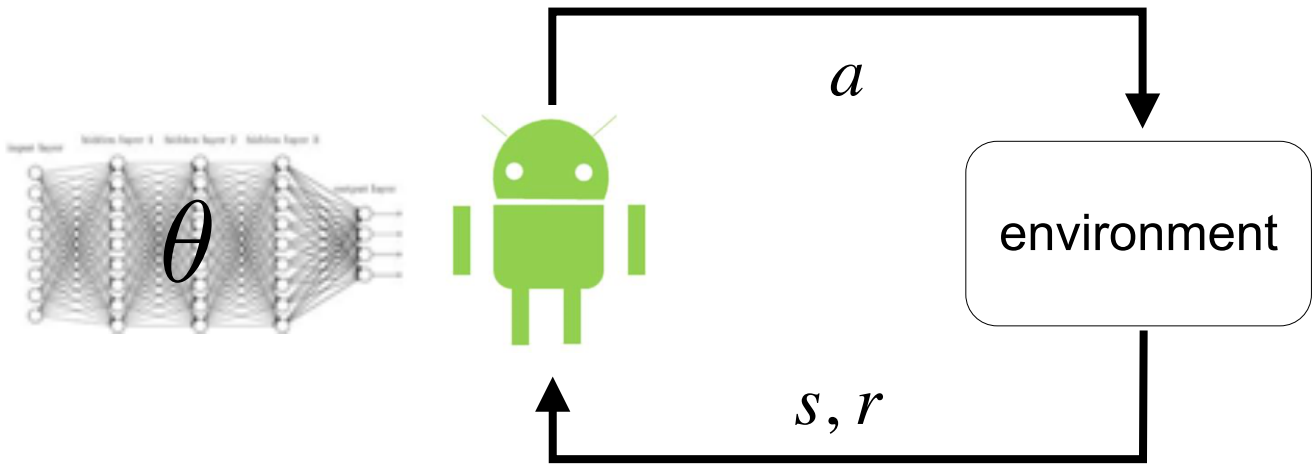
南京大学
人工智能学院
School of Artificial Intelligence, Nanjing University

Lecture 4
Markov Decision Process

Reinforcement learning



Agent's goal: learn a policy to maximize the return

T-step: $\sum_{t=1}^{T} r_t$ average: $\frac{1}{T}\sum_{t=1}^{T} r_t$ discounted: $\sum_{t=1}^{\infty} \gamma^t r_t$

Markov Process

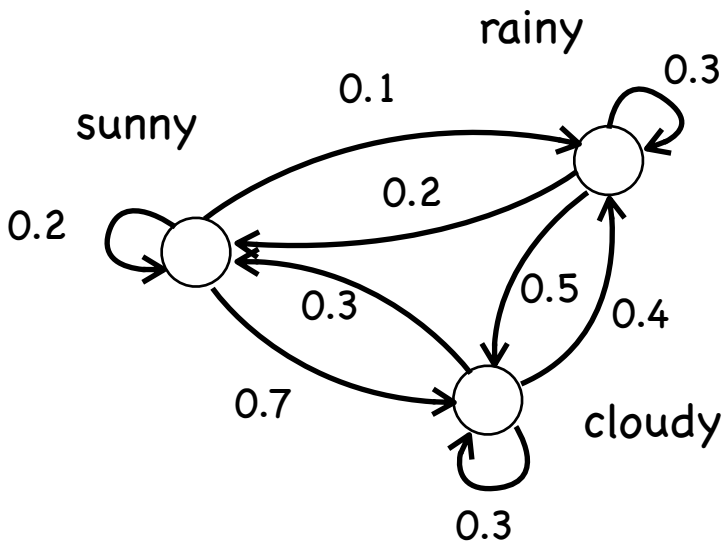
(finite) state space S, transition matrix P

a process $s_0, s_1, \ldots$ is Markov if

has no memory

$$P(s_{t+1} \mid s_t, \ldots, s_0) = P(s_{t+1} \mid s_t)$$

discrete S -> Markov chain

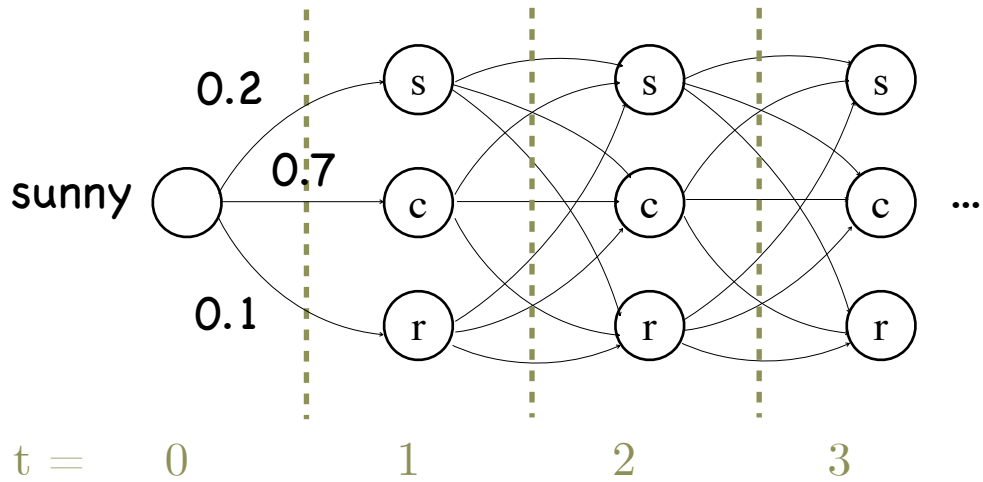


$$P = \begin{array}{c} \\ \text{sunny} \\ \text{cloudy} \\ \text{rainy} \end{array} \begin{array}{c} \begin{array}{ccc} \text{s} & \text{c} & \text{r} \end{array} \\ \begin{array}{|c|c|c|} \hline 0.2 & 0.7 & 0.1 \\ \hline 0.3 & 0.3 & 0.4 \\ \hline 0.2 & 0.5 & 0.3 \\ \hline \end{array} \end{array}$$

$$\boldsymbol{s}_{t+1} = \boldsymbol{s}_t P = \boldsymbol{s}_0 P^{t+1}$$

Markov Process

horizontal view



stationary distribution: $s == sP$

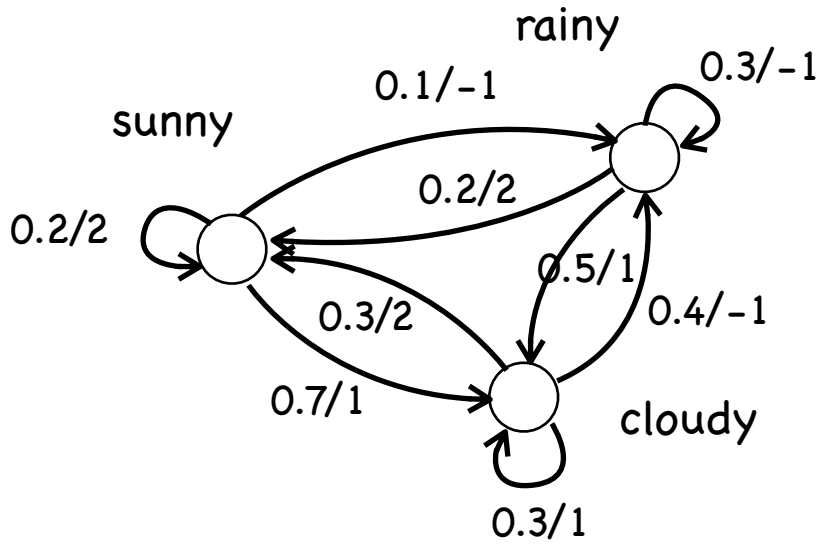
sampling from a Markov process:

s, c, c, r ...

s, c, s, c ...

Markov Reward Process

introduce reward function R



how to calculate the long-term total reward?

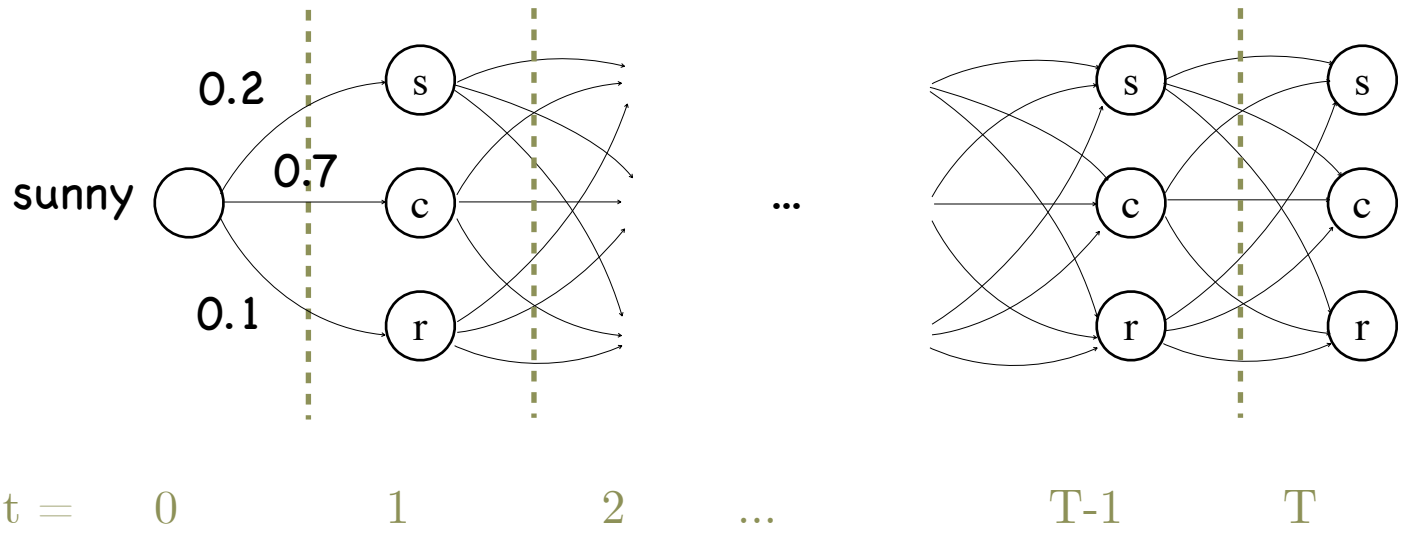
$$V(\text{sunny}) = E[\sum_{t=1}^{T} r_t | s_0 = \text{sunny}]$$

$$V(\text{sunny}) = E[\sum_{t=1}^{\infty} \gamma^t r_t | s_0 = \text{sunny}]$$

value function

Markov Reward Process

horizontal view: consider T steps

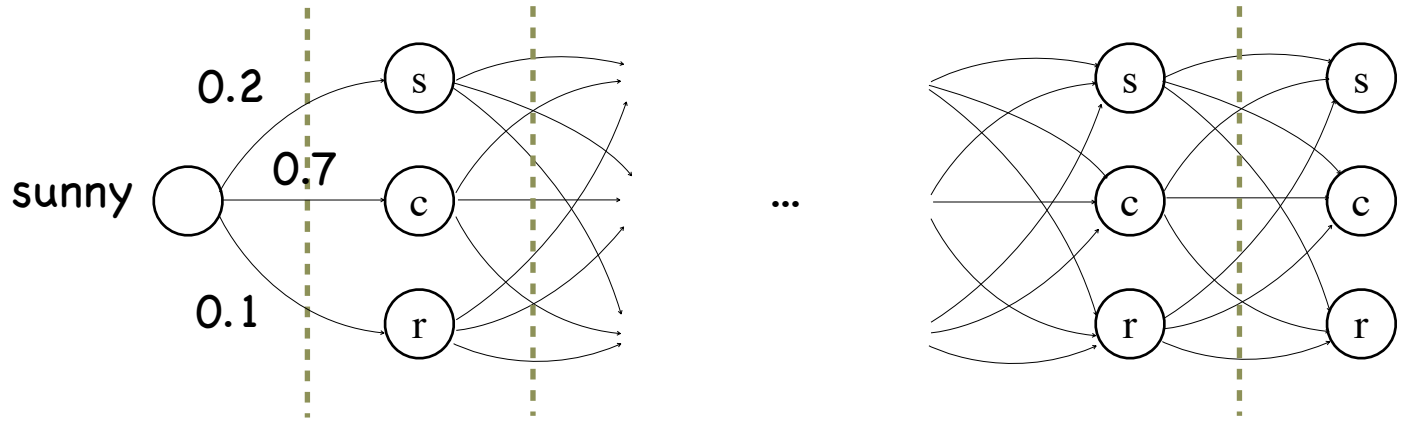


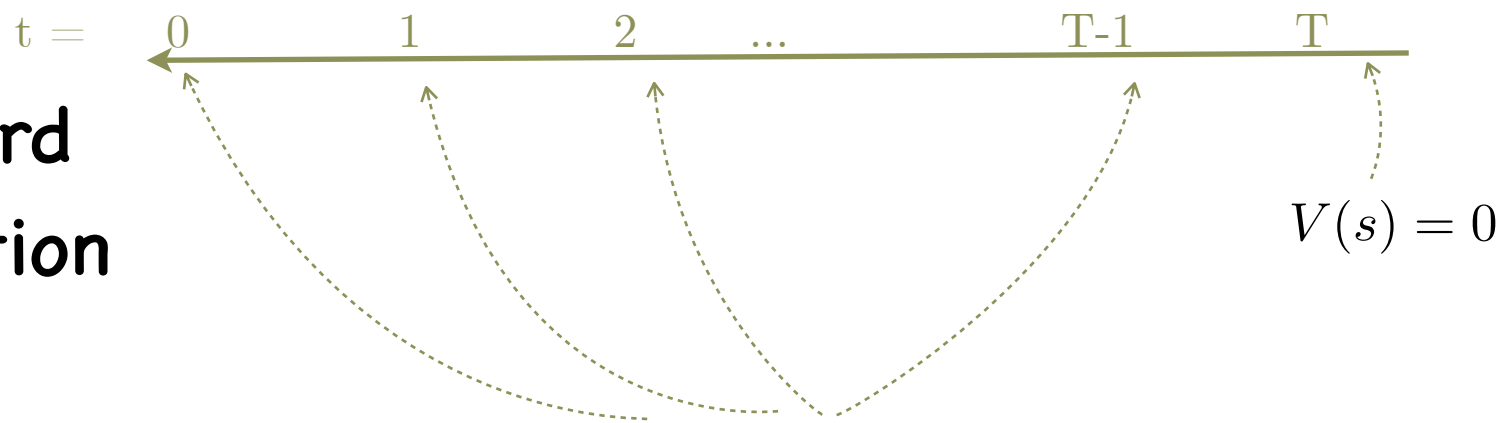
recursive definition:

$$
\begin{aligned}
V(\text{sunny}) =& P(\text{s}|\text{s})[R(\text{s}) + V(\text{s})] \\
&+ P(\text{c}|\text{s})[R(\text{c}) + V(\text{c})] \\
&+ P(\text{r}|\text{s})[R(\text{r}) + V(\text{r})]
\end{aligned}
\qquad
= \sum_{s} P(s|\text{sunny})\big(R(s) + V(s)\big)
$$

Markov Reward Process

horizontal view: consider T steps



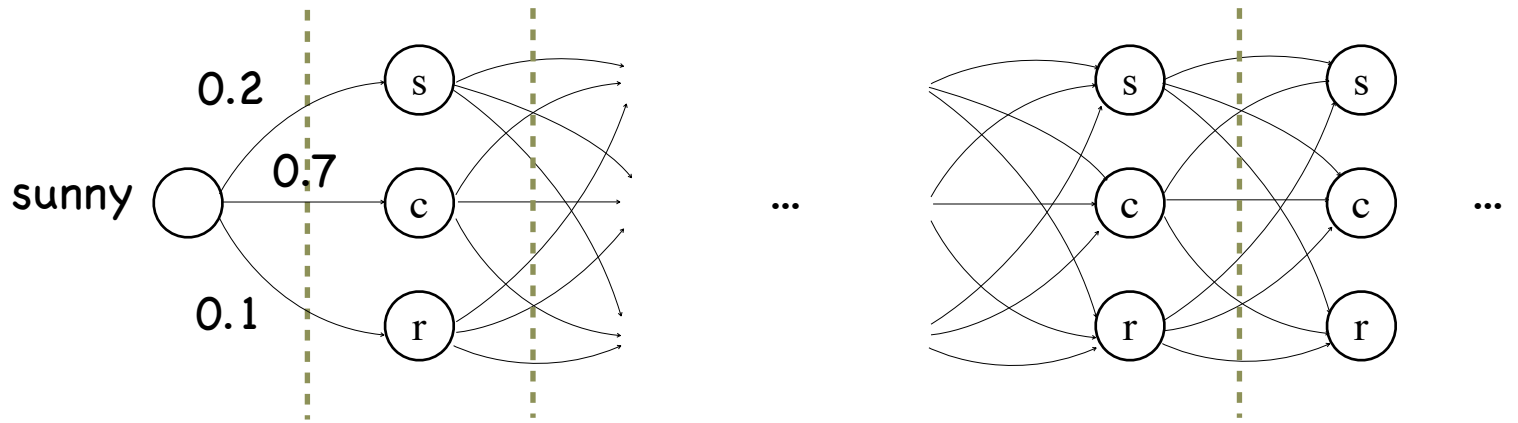


backward calculation

$$V(s) = \sum_{s'} P(s'|s)\big(R(s') + V(s')\big)$$

Markov Reward Process

horizontal view: consider discounted infinite steps



backward calculation

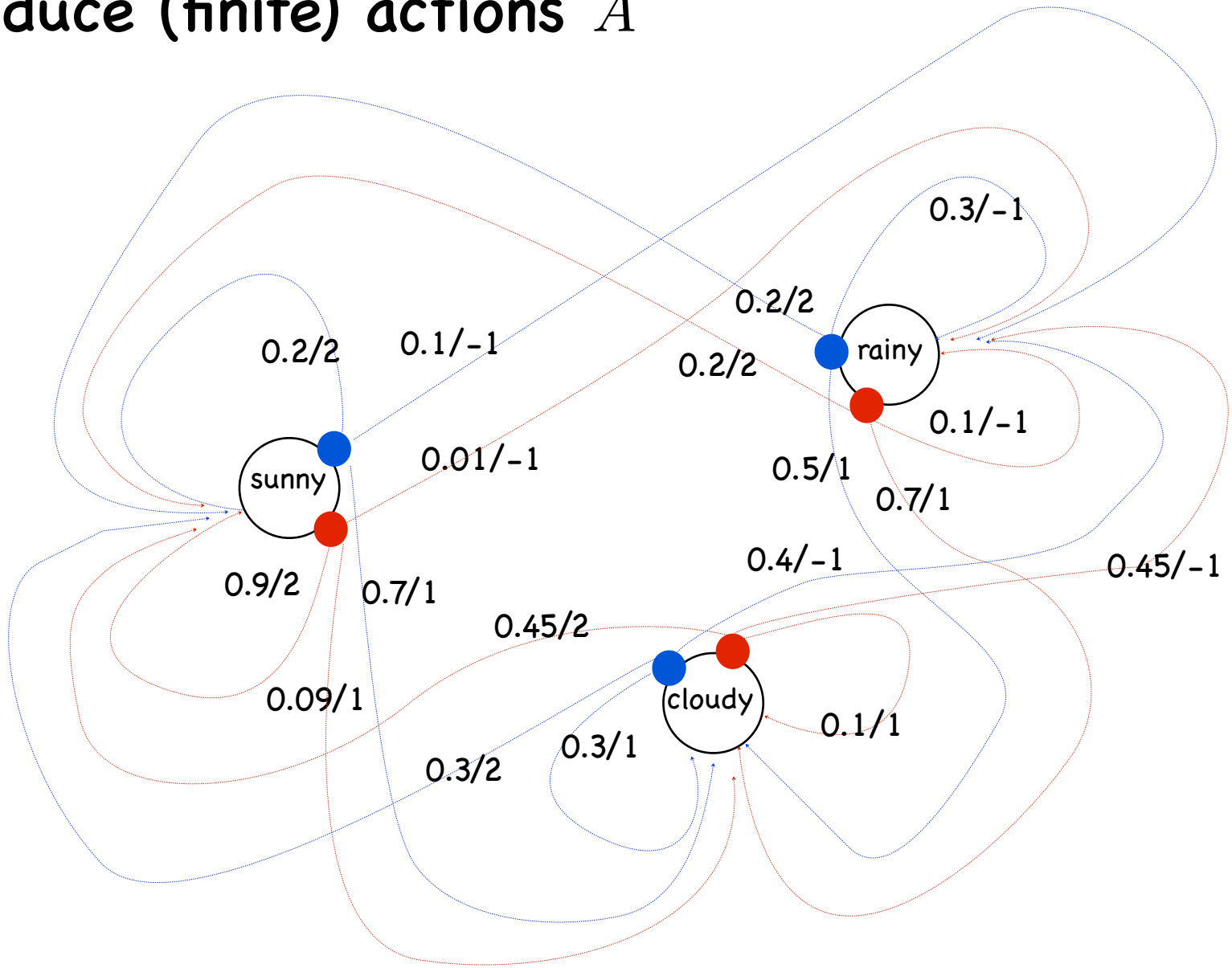
repeat until converges

$V(s) = 0$

$$V(s) = \sum_{s'} P(s'|s)\big(R(s') + \gamma V(s')\big)$$

Markov Decision Process

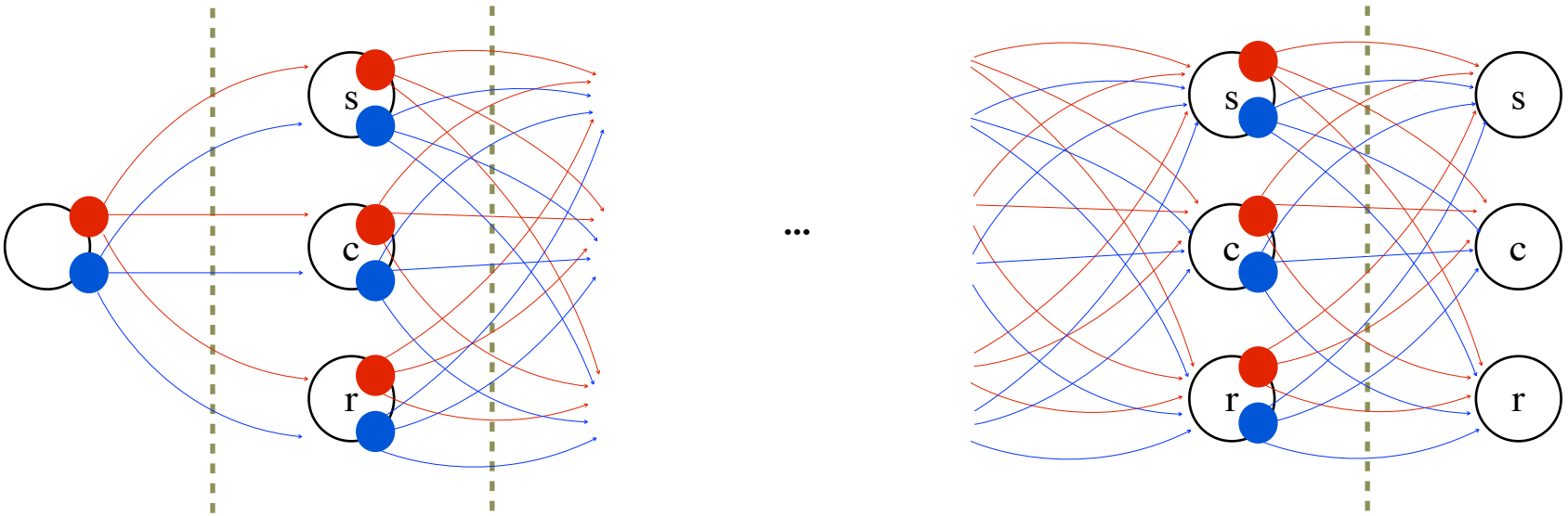
introduce (finite) actions A



Markov Decision Process

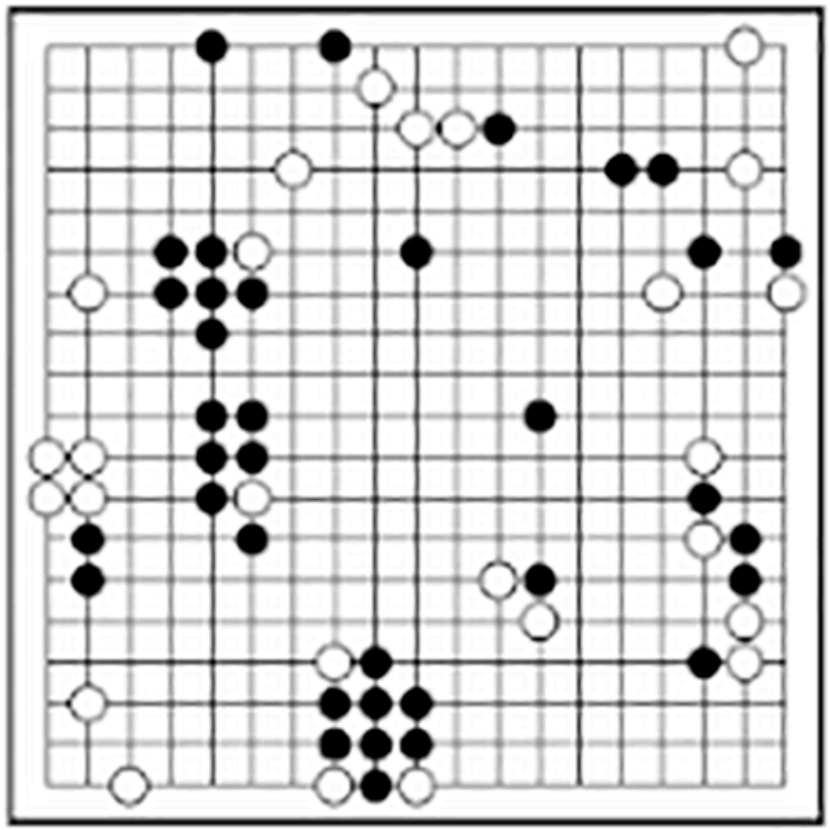
horizontal view

sunny



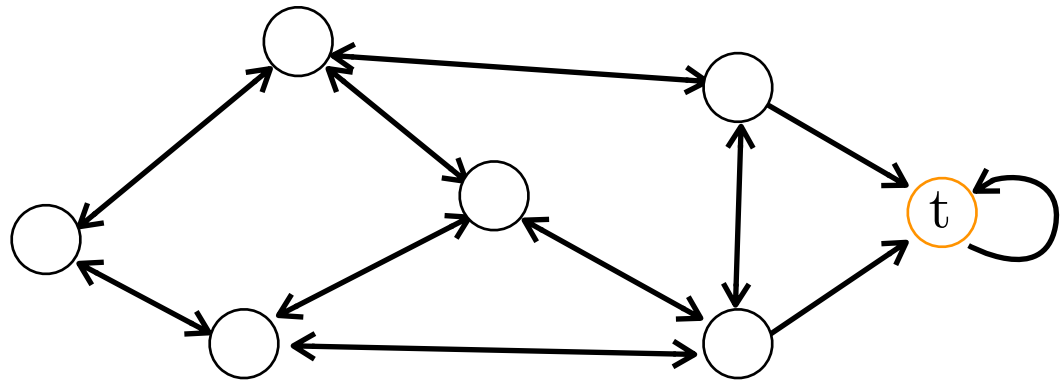
Markov Decision Process

horizontal view of the game of Go

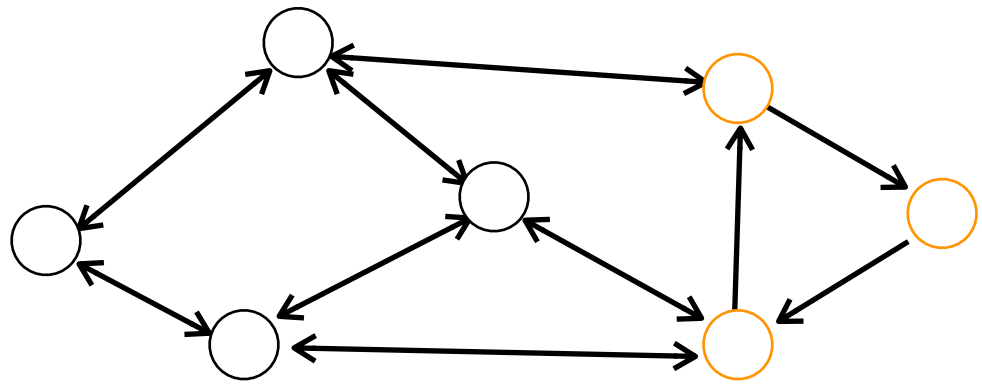


Markov Decision Process

goal-directed



stationary distribution

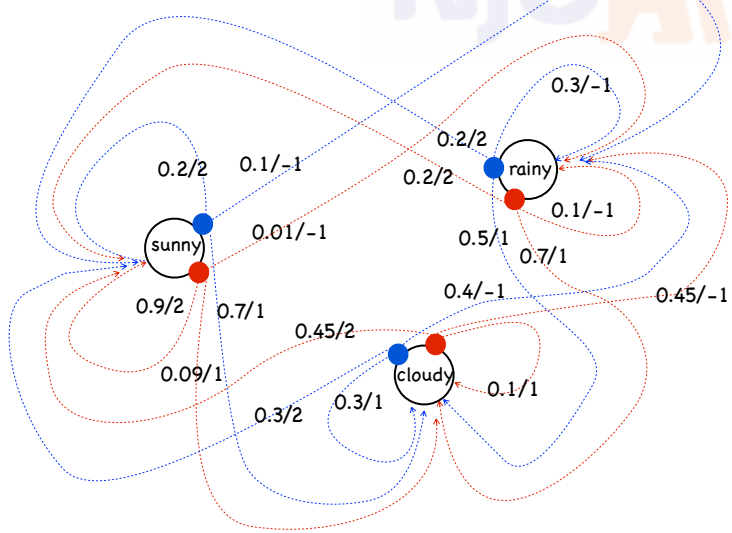


Markov Decision Process

MDP $<S, A, R, P>$ (often with γ)

essential model for RL

but not all of RL



policy

stochastic

$$\pi(a|s) = P(a|s)$$

deterministic

$$\pi(s) = \arg\max_a P(a|s)$$

$|A|^{|S|}$ deterministic policies

tabular representation

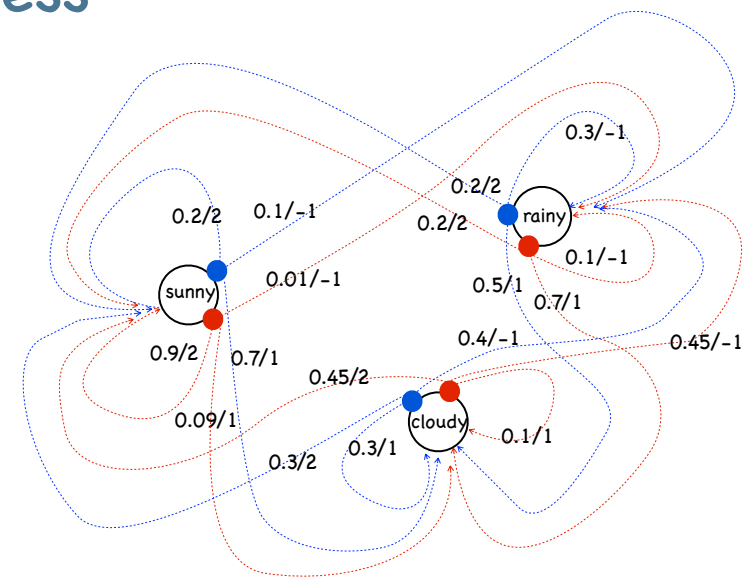
$\pi =$

s	0	0.3
	1	0.7
c	0	0.6
	1	0.4
r	0	0.1
	1	0.9

Expected return

how to calculate the expected total reward of a policy?

similar with the Markov Reward Process



MRP:

$$V(s) = \sum_{s'} P(s'|s)\big(R(s') + V(s')\big)$$

MDP:

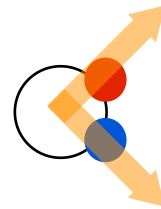
$$V^{\pi}(s) = \sum_{a} \pi(a|s) \sum_{s'} P(s'|s,a)\big(R(s,a,s') + V^{\pi}(s')\big)$$

expectation over actions

with respect to the policy

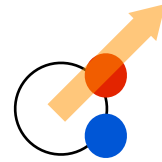
Q-function

state value function



$$V^{\pi}(s) = E[\sum_{t=1}^{T} r_t | s]$$

state-action value function



$$Q^{\pi}(s,a) = E[\sum_{t=1}^{T} r_t | s,a] = \sum_{s'} P(s'|s,a)\big(R(s,a,s') + V^{\pi}(s')\big)$$

consequently,

$$V^{\pi}(s) = \sum_{a} \pi(a|s) Q(s,a)$$

Q-function => policy

Optimality

s	0	0.3
	1	0.7
c	0	0.6
	1	0.4
r	0	0.1
	1	0.9

there exists an optimal policy π^*

$$\forall \pi, \forall s, V^{\pi^*}(s) \geq V^{\pi}(s)$$

optimal value function

$$\forall s, V^*(s) = V^{\pi^*}(s)$$

$$\forall s, \forall a, Q^*(s,a) = Q^{\pi^*}(s,a)$$

Bellman optimality equations

s	0	0.3
	1	0.7
c	0	0.6
	1	0.4
r	0	0.1
	1	0.9

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

from the relation between V and Q

$$Q^*(s,a) = \sum_{s'} P(s'|s,a)\big(R(s,a,s') + \gamma V^*(s')\big)$$

we have

$$Q^*(s,a) = \sum_{s'} P(s'|s,a)\big(R(s,a,s') + \gamma \max_a Q^*(s',a)\big)$$

$$V^*(s) = \max_a \sum_{s'} P(s'|s,a)\big(R(s,a,s') + \gamma V^*(s')\big)$$

the unique fixed point is the optimal value function

Solve optimal policy in MDP

idea:

how is the current policy — policy evaluation

improve the current policy — policy improvement

policy evaluation: backward calculation

$$V^{\pi}(s) = \sum_{a} \pi(a|s) \sum_{s'} P(s'|s,a)\big(R(s,a,s') + \gamma V^{\pi}(s')\big)$$

policy improvement: from the Bellman optimality equation

$$V(s) \leftarrow \max_{a} Q^{\pi}(s,a)$$

Solve optimal policy in MDP

policy improvement: from the Bellman optimality equation

$$V(s) \leftarrow \max_a Q^{\pi}(s,a)$$

let π' be derived from this update

$$\begin{aligned} V^{\pi}(s) &\leq Q^{\pi}(s,\pi'(s)) \\ &= \sum\nolimits_{s'} P(s'|s,\pi'(s))(R(s,\pi'(s),s') + \gamma V^{\pi}(s')) \\ &\leq \sum\nolimits_{s'} P(s'|s,\pi'(s))(R(s,\pi'(s),s') + \gamma Q^{\pi}(s',\pi'(s))) \\ &= \dots \\ &= V^{\pi'} \end{aligned}$$

so the policy is improved

Solve optimal policy in MDP

Policy iteration algorithm:

loop until converges

policy evaluation: calculate V

policy improvement: choose the action greedily

$$\pi_{t+1}(s) = \arg\max_a Q^{\pi_t}(s,a)$$

converges: $V^{\pi_{t+1}}(s) = V^{\pi_t}(s)$

$$Q^{\pi_{t+1}}(s,a) = \sum_{s'} P(s'|s,a)\big(R(s,a,s') + \gamma \max_a Q^{\pi_t}(s',a)\big)$$

recall the optimal value function about Q

Solve optimal policy in MDP

embed the policy improvement in evaluation

Value iteration algorithm:

$V_0 = 0$

for t=0, 1, ...

  for all s <- synchronous v.s. asynchronous

  $V_{t+1}(s) = \max_a \sum_{s'} P(s'|s,a)\big(R(s,a,s') + \gamma V_t(s)\big)$

  end for

  break if $||V_{t+1} - V_t||_\infty$ is small enough

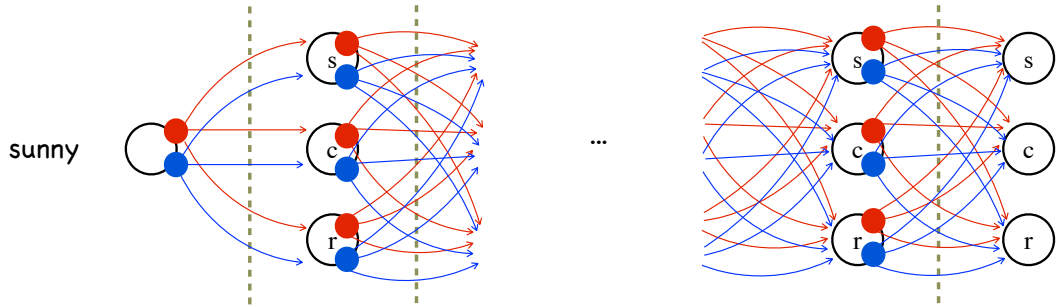
end for

recall the optimal value function about V

Solve optimal policy in MDP

$$Q^{\pi_{t+1}}(s,a) = \sum_{s'} P(s'|s,a)\big(R(s,a,s') + \gamma \max_a Q^{\pi_t}(s',a)\big)$$

$$V_{t+1}(s) = \max_a \sum_{s'} P(s'|s,a)\big(R(s,a,s') + \gamma V_t(s')\big)$$



R. E. Bellman
1920-1984

Dynamic programming

Complexity

needs $\Theta(|S| \cdot |A|)$ iterations to converge on deterministic MDP

[O. Madani. Polynomial Value Iteration Algorithms for Deterministic MDPs. UAI'02]

curse of dimensionality: Go board 19x19, $|S|=2.08\times10^{170}$

[https://github.com/tromp/golegal]