南京大學
NANJING UNIVERSITY

SKL
国家重点实验室
STATE KEY LABORATORY

LAMDA
Learning And Mining from DatA
http://lamda.nju.edu.cn

强化学习前沿

俞扬

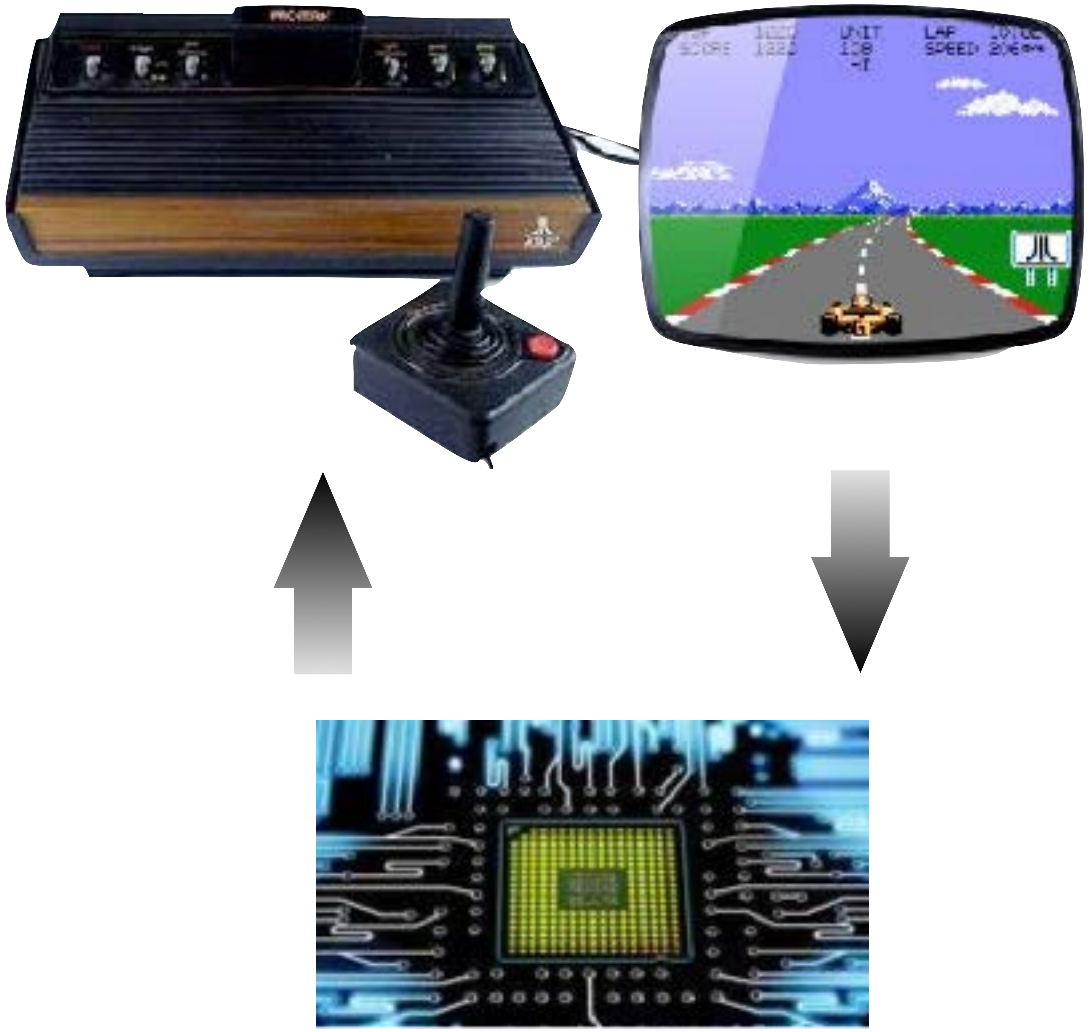
南京大学
软件新技术国家重点实验室
机器学习与数据挖掘研究所

latest slides: http://lamda.nju.edu.cn/yuy/adl-rl.ashx

The Atari games

Deepmind Deep Q-learning on Atari

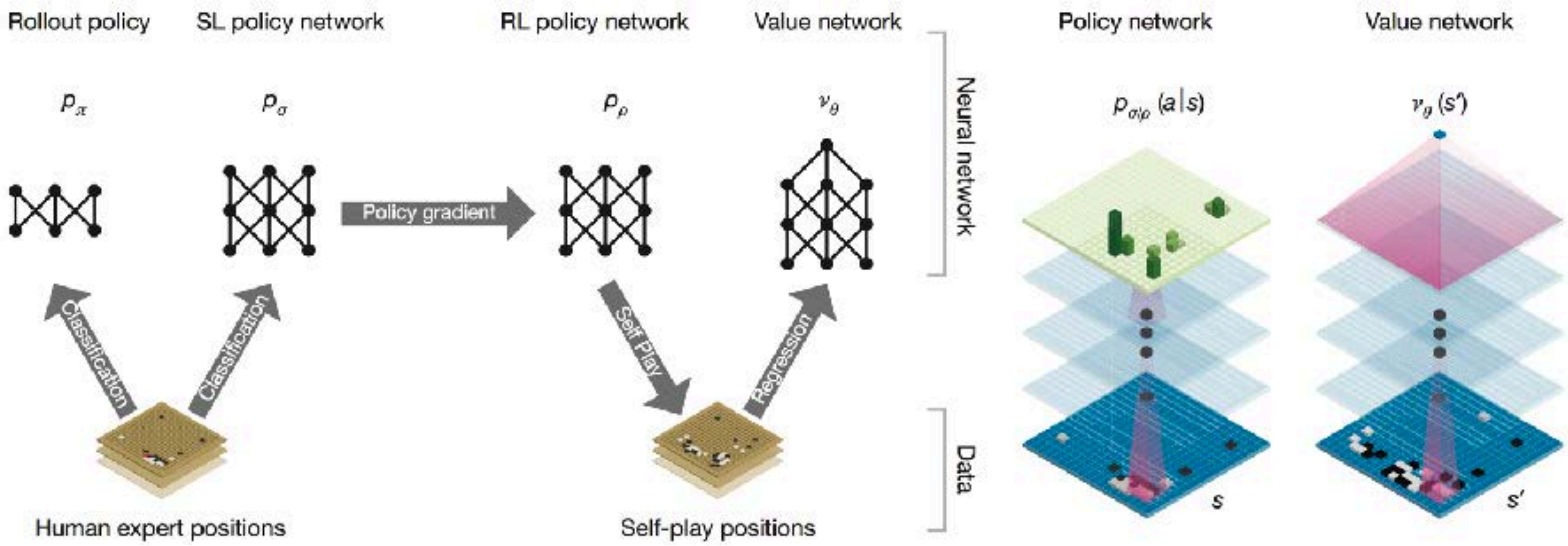
[Mnih *et al*. Human-level control through deep reinforcement learning. Nature, 518(7540): 529-533, 2015]



The game of Go

Deepmind AlphaGo system

[Silver *et al*. Mastering the game of Go with deep neural networks and tree search. Nature, 529(7587): 484−489, 2016.]



Aims

in the following 3 hours

1. what is reinforcement learning (RL)
2. what does RL capable of
3. principles of RL algorithms
4. some directions of RL

Outline

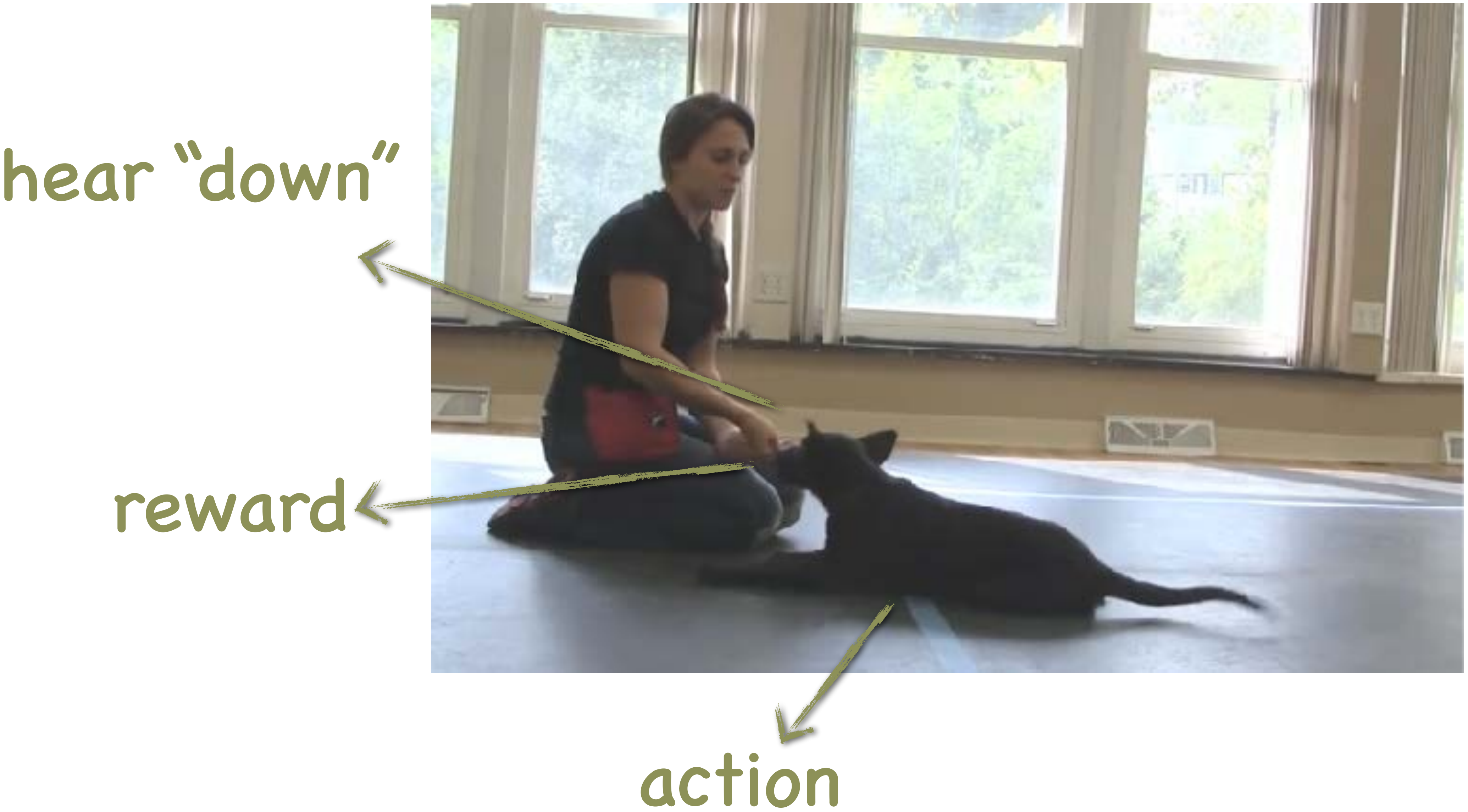
- ✦ Introduction
- ✦ Markov Decision Process
- ✦ From MDP to Reinforcement Learning
- ✦ Function Approximation
- ✦ Policy Search
- ✦ Deep Reinforcement Learning

How to train a dog?

PHASE 1

DOWN

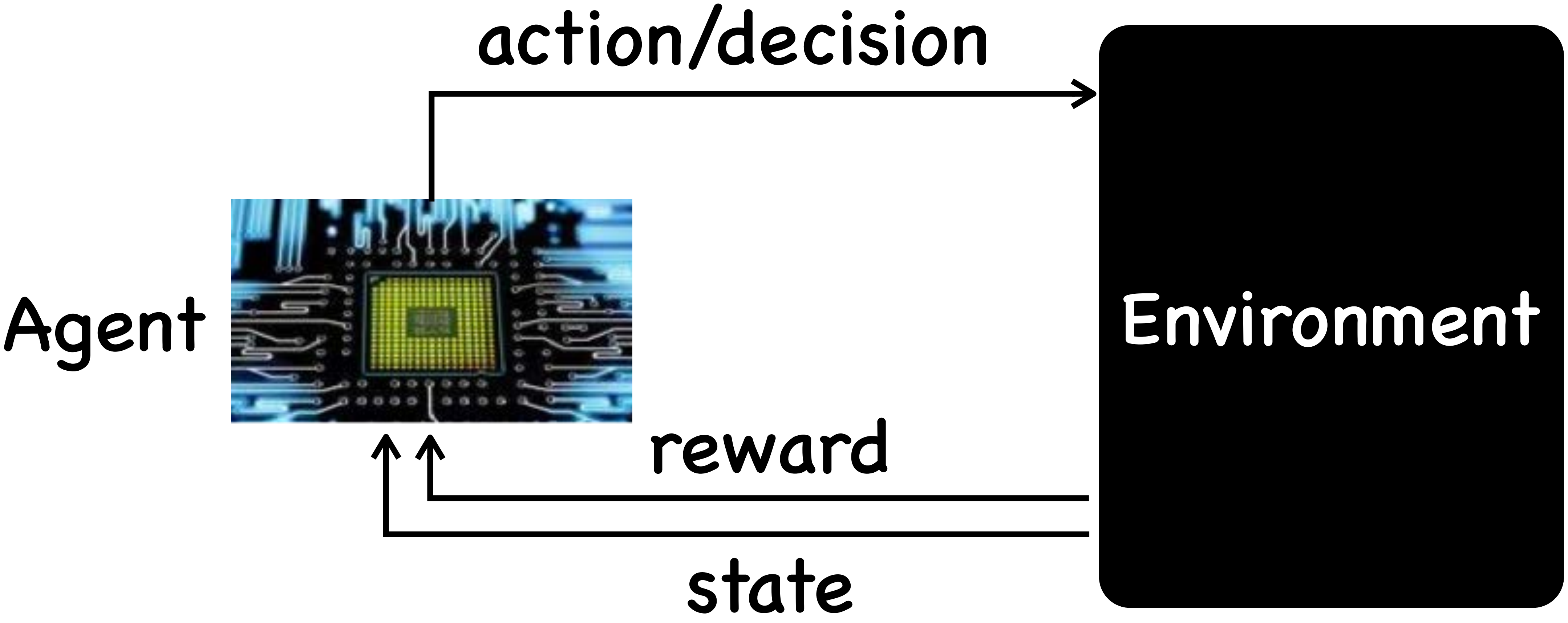
How to train a dog?



dog learns from rewards to adapt to the environment

can computers do similarly?

Reinforcement learning setting



action/decision

Agent

Environment

reward

state

$$<A,\ S,\ R,\ P>$$

Action space: A

State space: S

Reward: $R : S \times A \times S \to \mathbb{R}$

Transition: $P : S \times A \to S$

Reinforcement learning setting

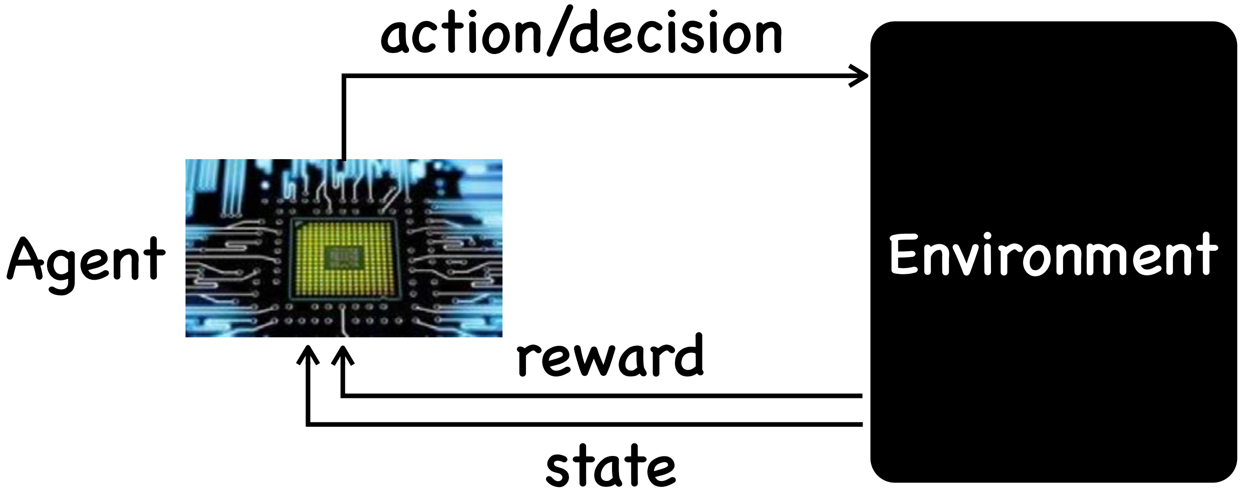
$$<A,\ S,\ R,\ P>$$

Action space: A

State space: S

Reward: $R : S \times A \times S \to \mathbb{R}$

Transition: $P : S \times A \to S$



Agent:

Policy: $\pi : S \times A \to \mathbb{R}, \quad \sum_{a \in A} \pi(a|s) = 1$

Policy (deterministic): $\pi : S \to A$

Agent's view:

$$s_0,\ \underset{\overset{\uparrow}{\pi(s_0)}}{a_0},\ r_1, s_1,\ \underset{\overset{\uparrow}{\pi(s_1)}}{a_2},\ r_2, s_2,\ \underset{\overset{\uparrow}{\pi(s_2)}}{a_3},\ r_3, s_3, \ldots$$

Reinforcement learning setting

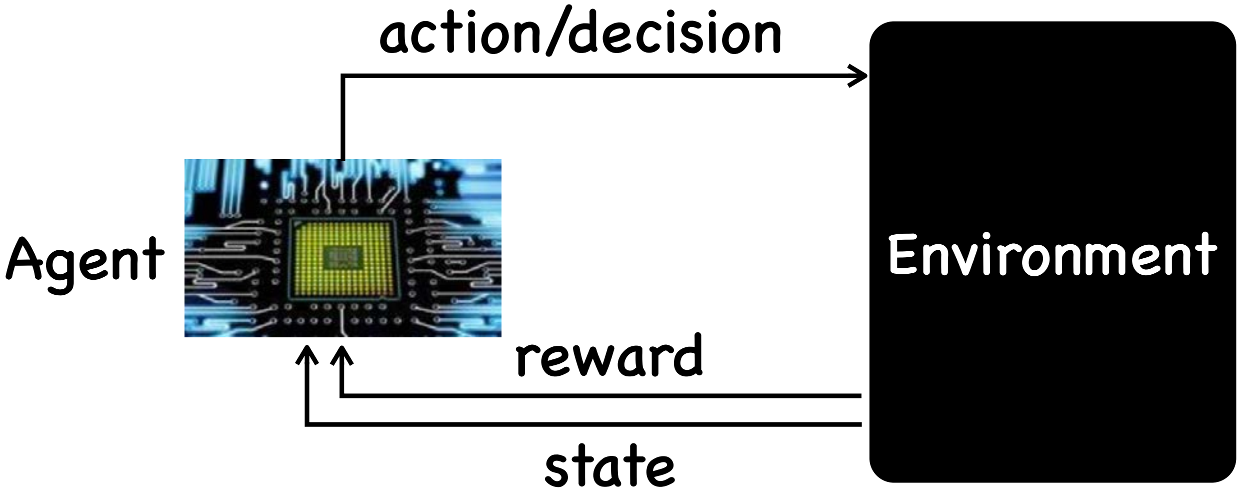
$<A, S, R, P>$

Action space: A

State space: S

Reward: $R : S \times A \times S \to \mathbb{R}$

Transition: $P : S \times A \to S$



Agent: Policy: $\pi : S \times A \to \mathbb{R}, \quad \sum_{a\in A} \pi(a|s) = 1$

Policy (deterministic): $\pi : S \to A$

Agent's goal:

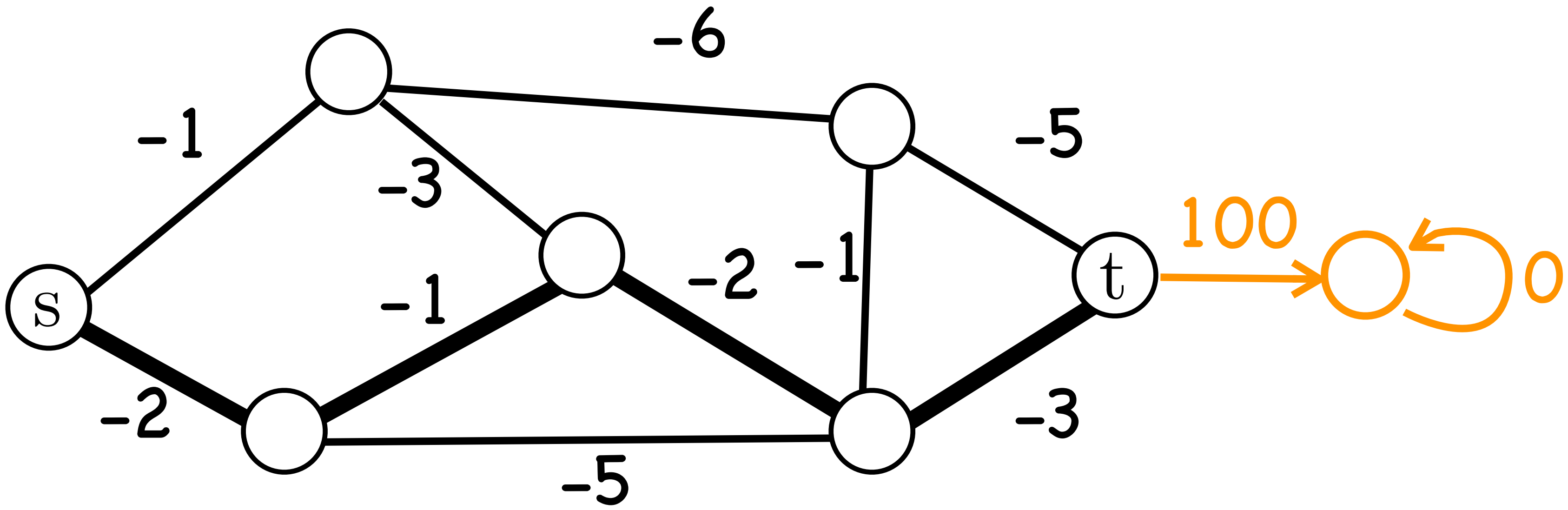
learn a policy to maximize long-term total reward

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

all RL tasks can be defined by maximizing total reward

Reward examples

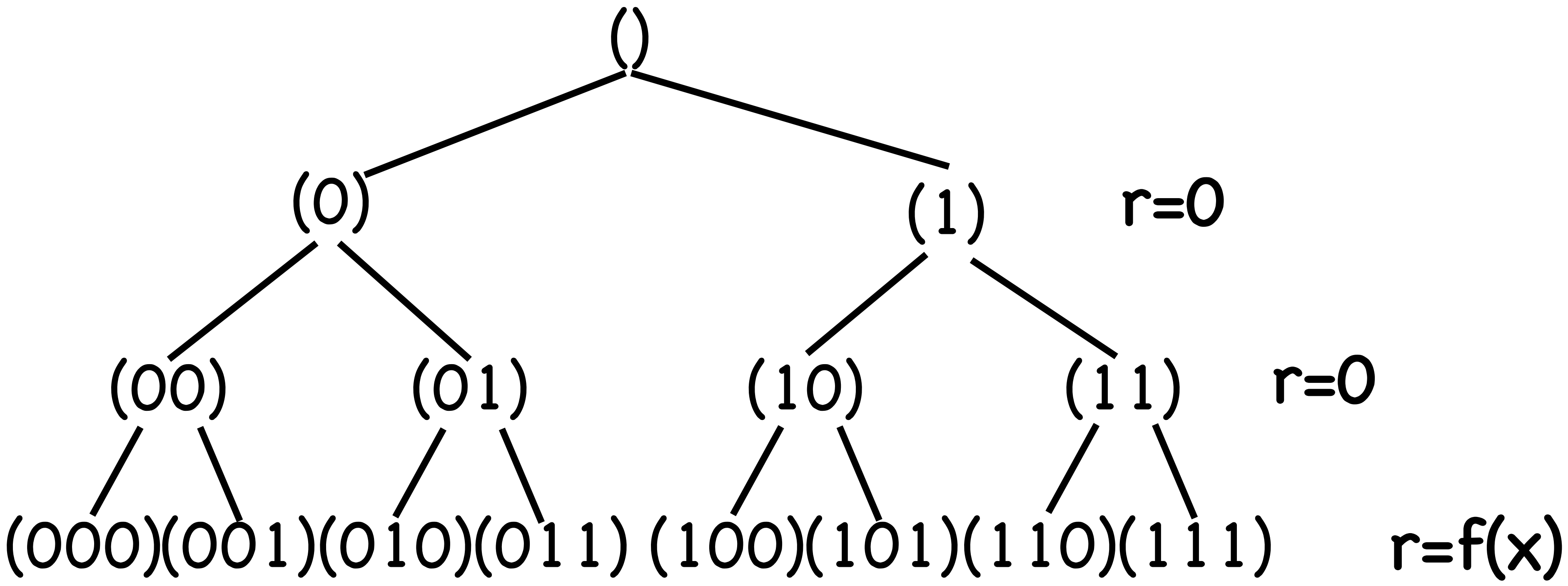
shortest path:



- every node is a state, an action is an edge out
- reward function = the negative edge weight
- optimal policy leads to the shortest path

Reward examples

general binary space problem $\max\limits_{x \in \{0,1\}^n} f(x)$



solving the optimal policy is NP-hard!

Difference between RL and planning?

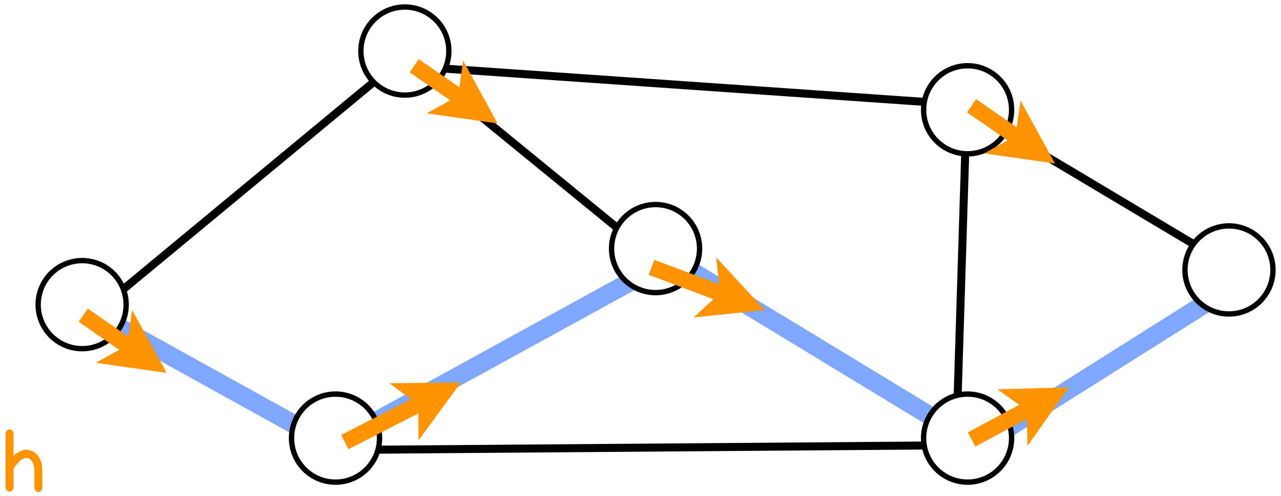
what if we use planning/search methods to find actions that maximize total reward

Planing: find an optimal solution

RL: find an optimal policy from samples

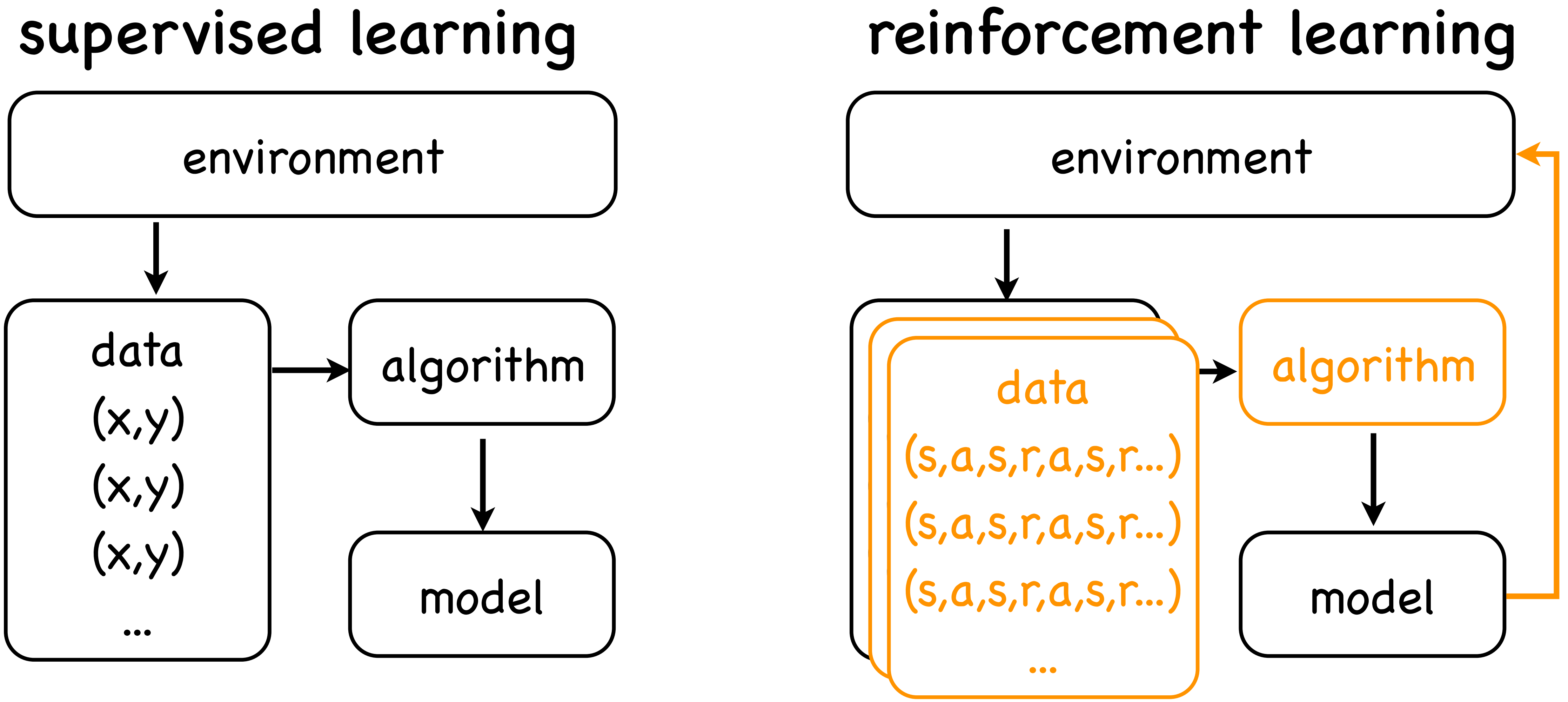
planning: shortest-path

RL: shortest-path policy without knowing the graph



Difference between RL and SL?

supervised learning also learns a model …



learning from labeled data

open loop

passive data

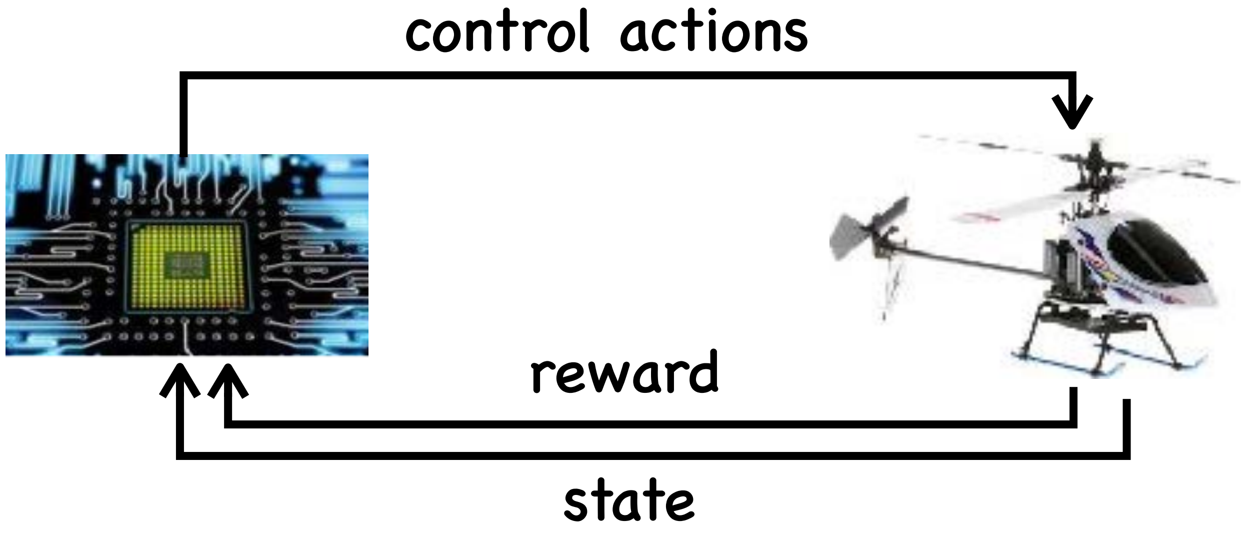
learning from delayed reward

closed loop

explore environment

Applications

learning robot skills



https://www.youtube.com/watch?v=VCdxqn0fcnE

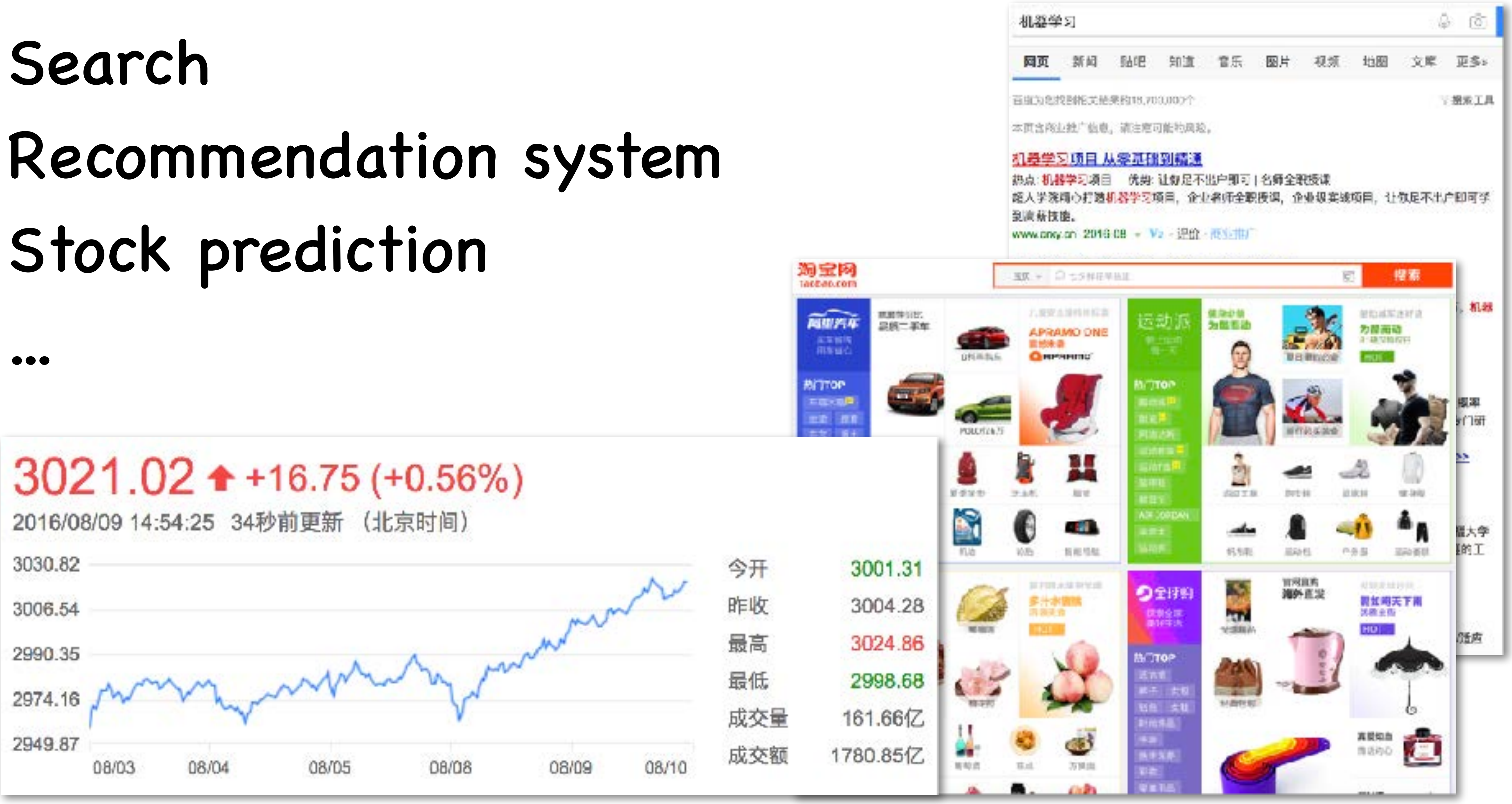
More applications

Search

Recommendation system

Stock prediction

...



3021.02 ↑ +16.75 (+0.56%)

2016/08/09 14:54:25 34秒前更新 （北京时间）

今开	3001.31
昨收	3004.28
最高	3024.86
最低	2998.68
成交量	161.66亿
成交额	1780.85亿

every decision changes the world

Markov Decision Process

essential mathematical model for RL

Markov Process

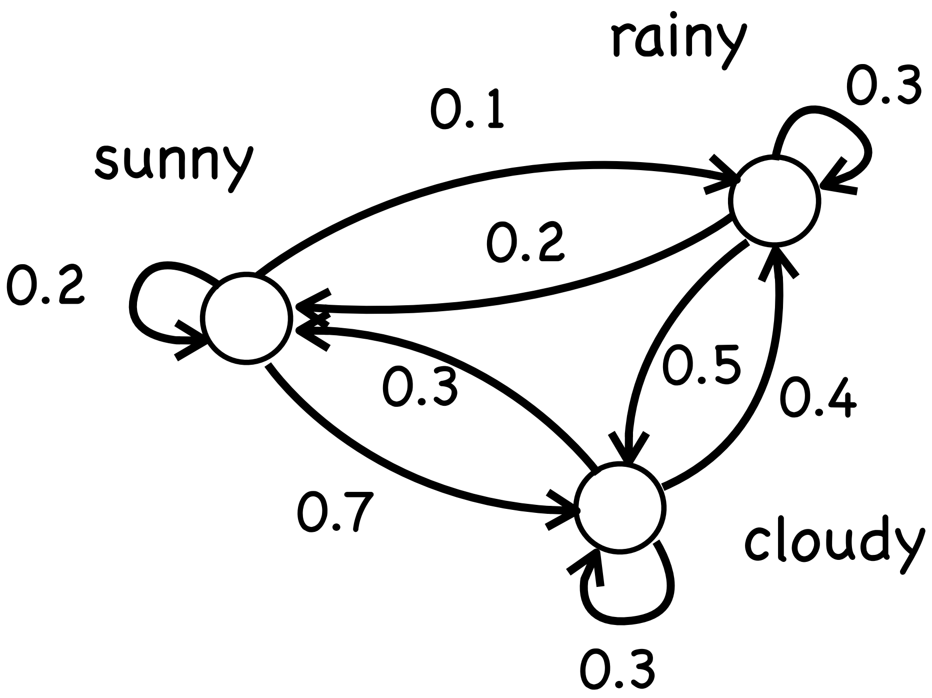
(finite) state space S, transition matrix P

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

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

has no memory

discrete S -> Markov chain

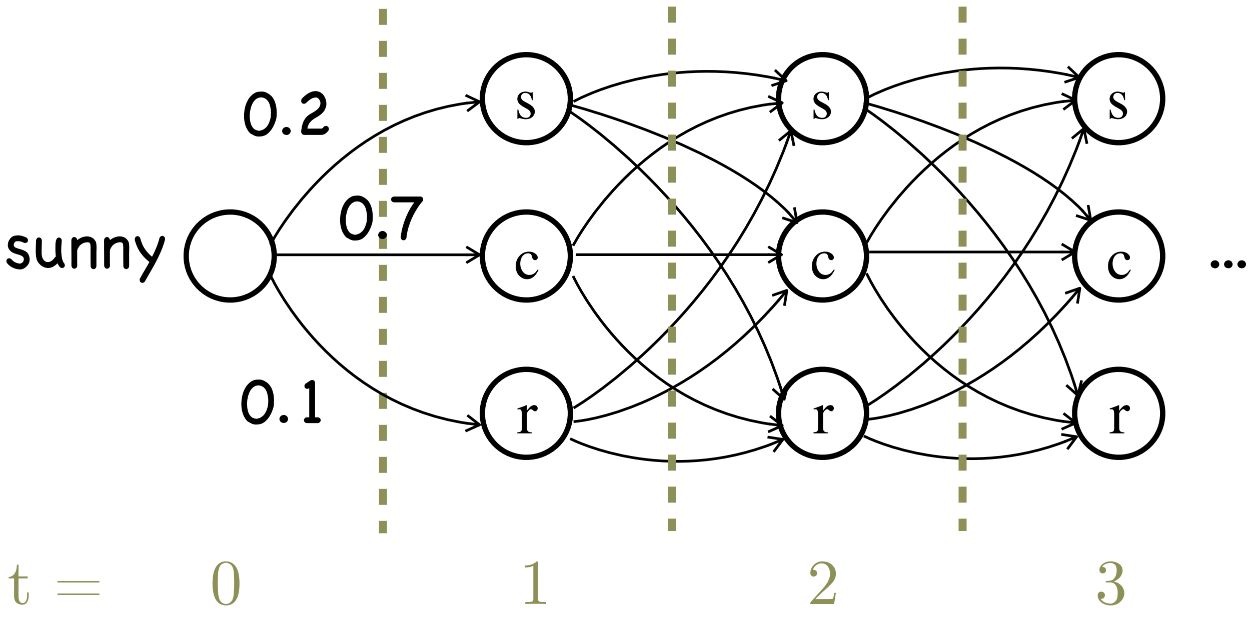


$$P = \begin{array}{c|c|c|c|} & \text{s} & \text{c} & \text{r} \\ \hline \text{sunny} & 0.2 & 0.7 & 0.1 \\ \hline \text{cloudy} & 0.3 & 0.3 & 0.4 \\ \hline \text{rainy} & 0.2 & 0.5 & 0.3 \\ \hline \end{array}$$

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

Markov Process

horizontal view



stationary distribution: $s == sP$

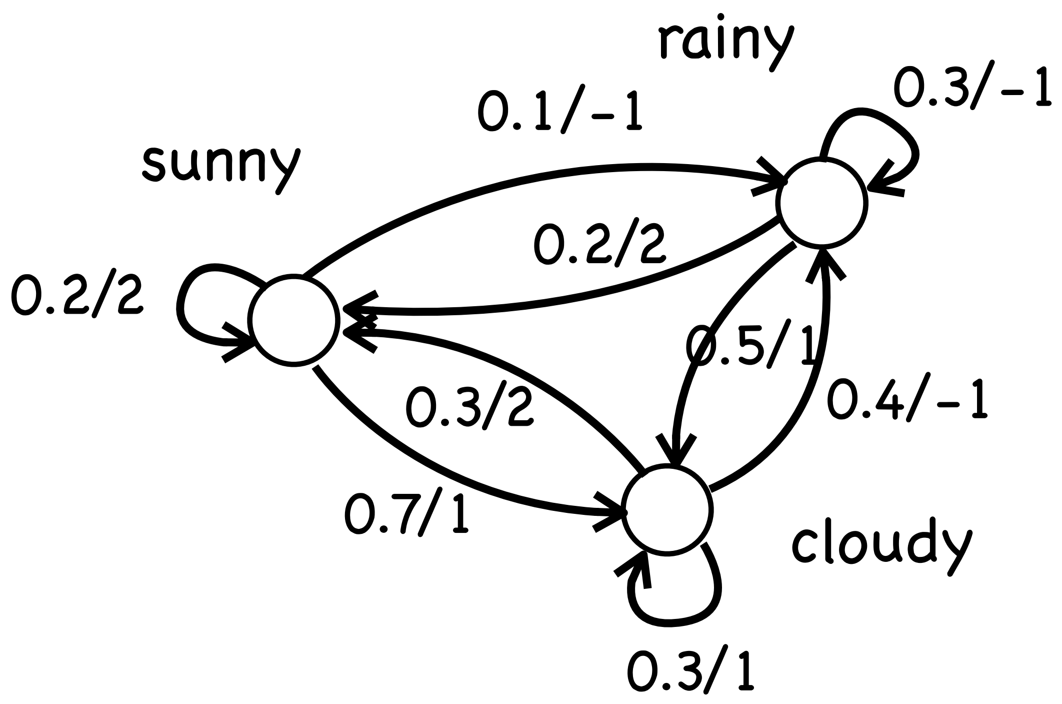
sampling from a Markov process:

s, c, c, r ...

s, c, s, c ...

Markov Reward Process

introduce reward function R



how to calculate the long-term total reward?

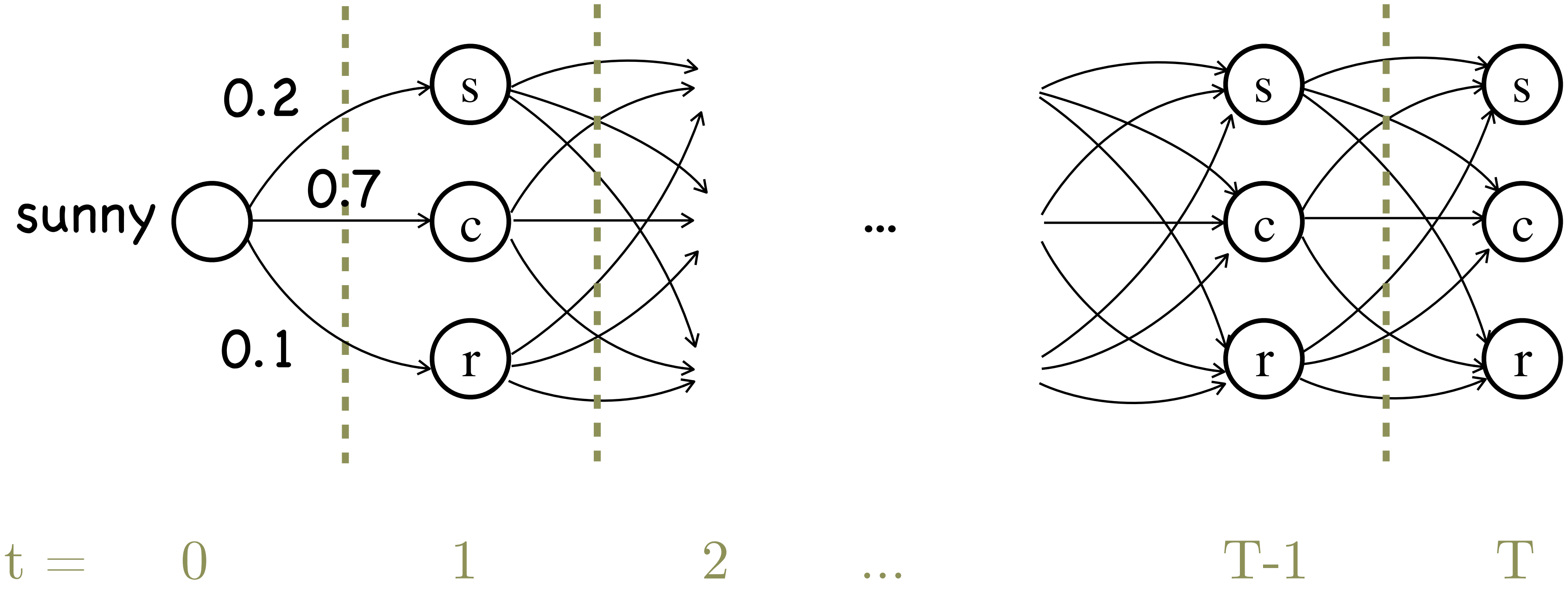
$$V(\text{sunny}) = E[\sum_{t=1}^{T} r_t | s_0 = \text{sunny}]$$

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

value function

Markov Reward Process

horizontal view: consider T steps

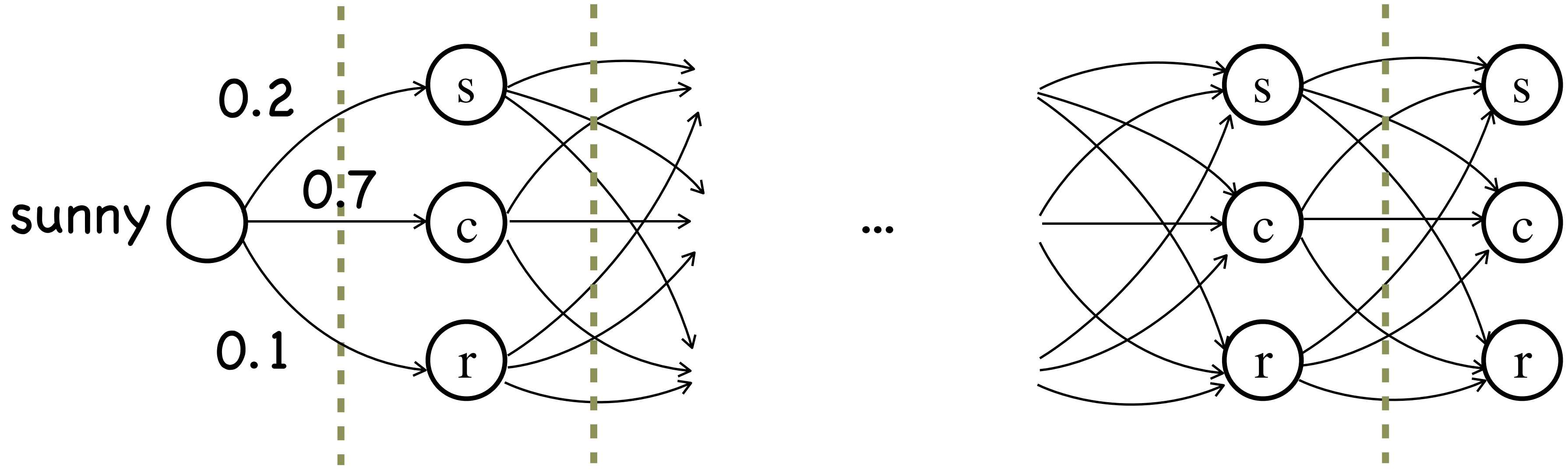


recursive definition:

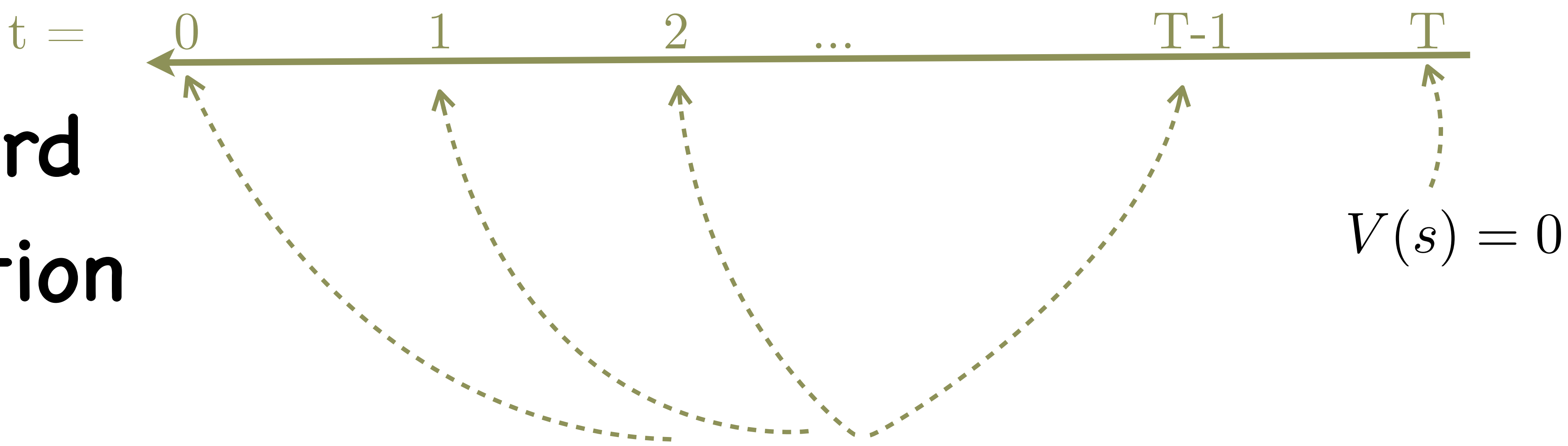
$$
\begin{aligned}
V(\mathrm{sunny}) =& P(\mathrm{s}|\mathrm{s})[R(\mathrm{s}) + V(\mathrm{s})] \\
&+ P(\mathrm{c}|\mathrm{s})[R(\mathrm{c}) + V(\mathrm{c})] \\
&+ P(\mathrm{r}|\mathrm{s})[R(\mathrm{r}) + V(\mathrm{r})]
\end{aligned}
\qquad = \sum_{s} P(s|\mathrm{sunny})\big(R(s) + V(s)\big)
$$

Markov Reward Process

horizontal view: consider T steps



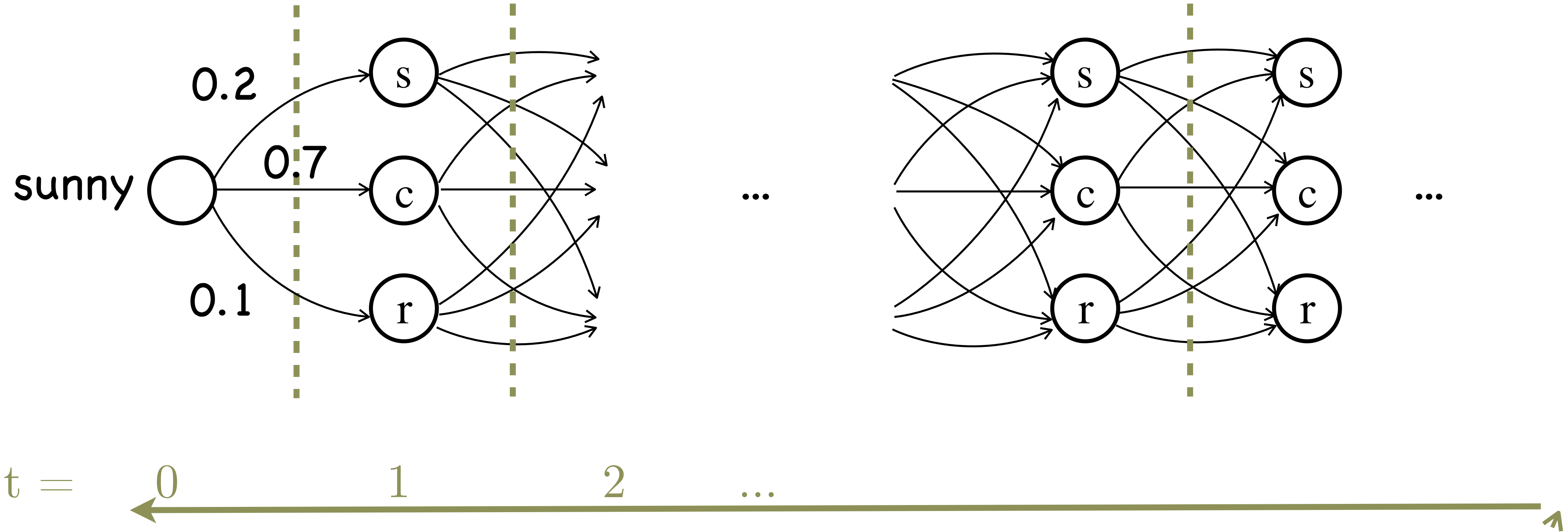
backward calculation



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

Markov Reward Process

horizontal view: consider discounted infinite steps



backward calculation

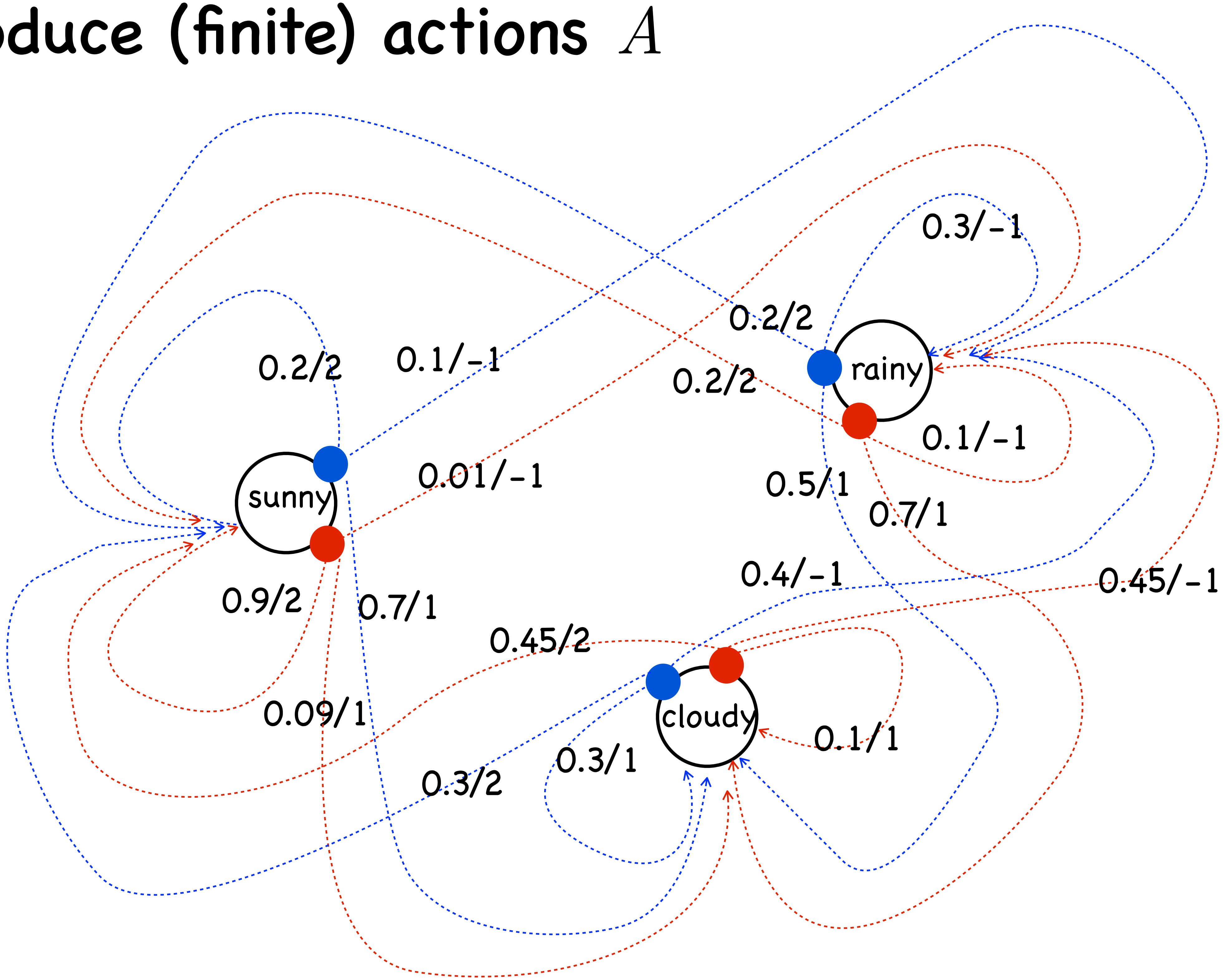
repeat until converges

$V(s) = 0$

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

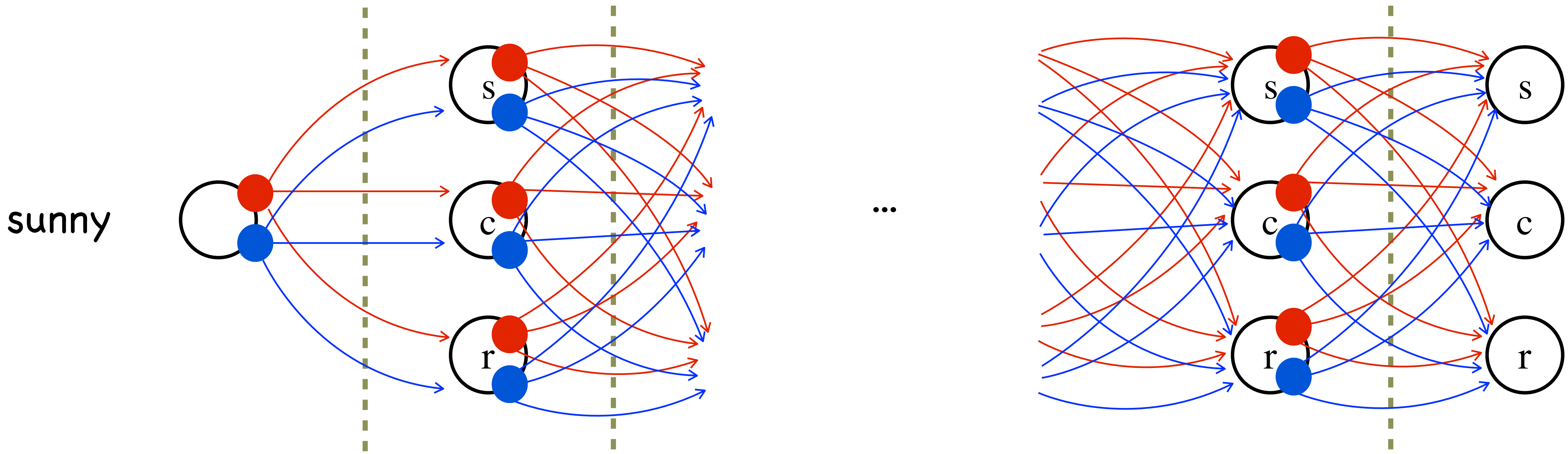
Markov Decision Process

introduce (finite) actions A



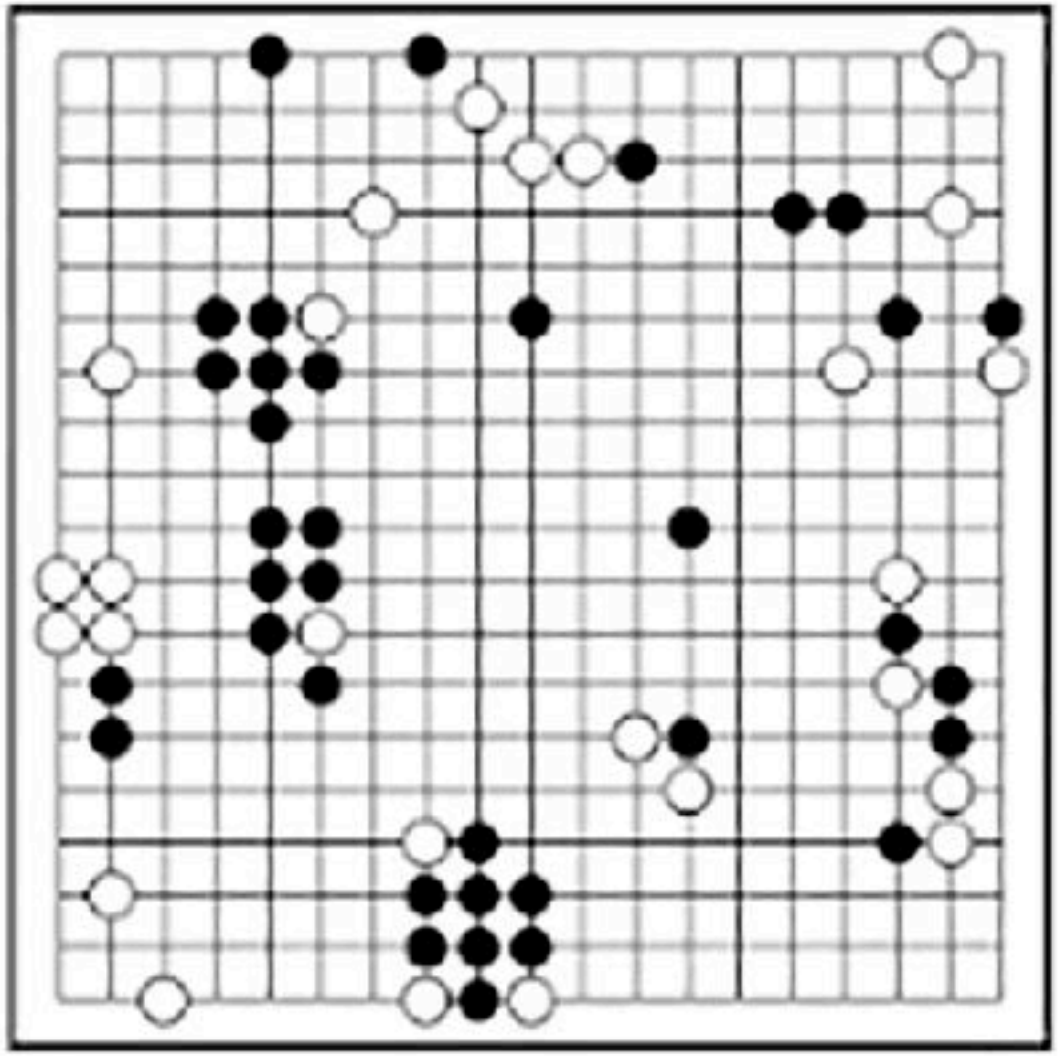
Markov Decision Process

horizontal view



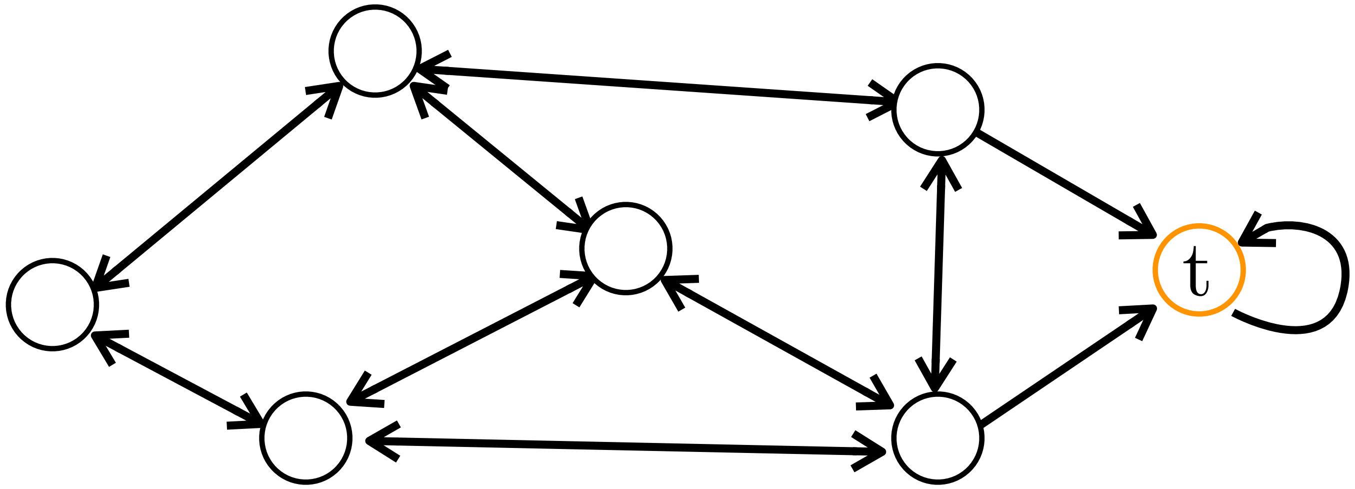
Markov Decision Process

horizontal view of the game of Go

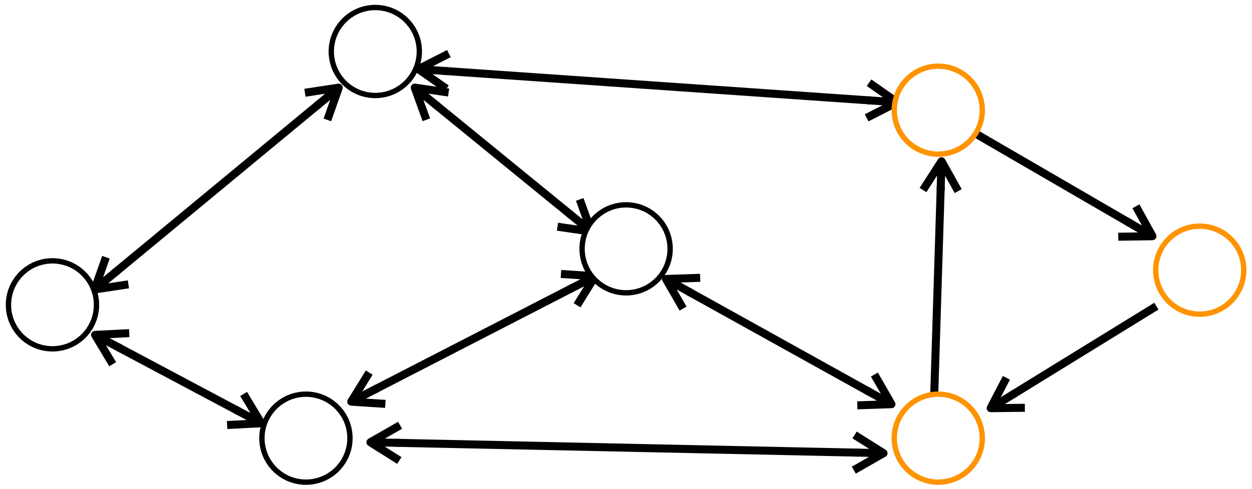


Markov Decision Process

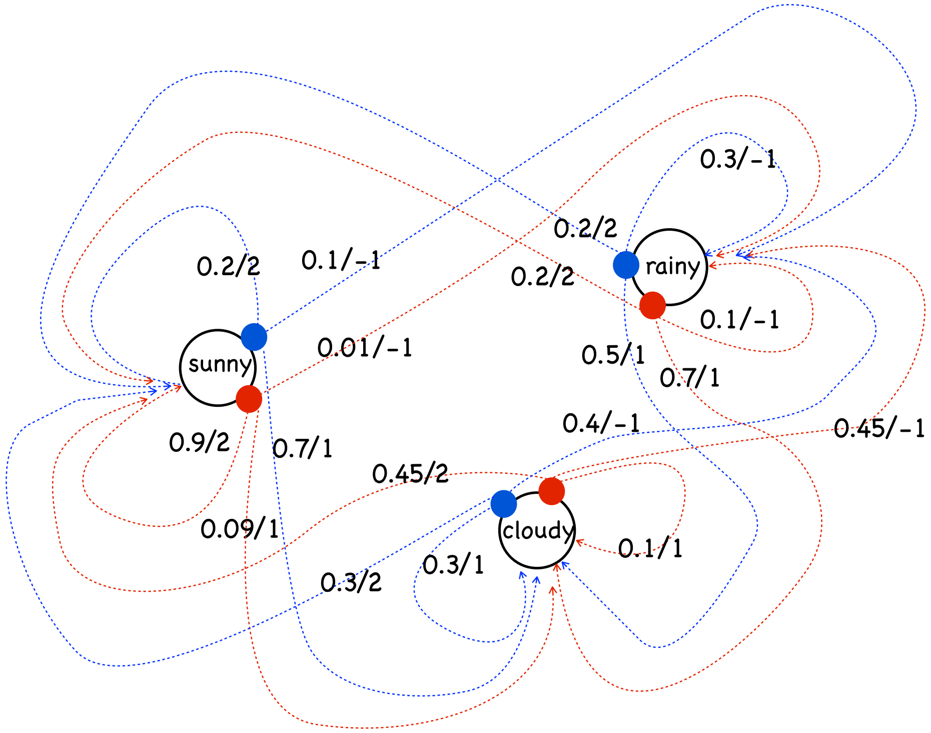
goal-directed



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

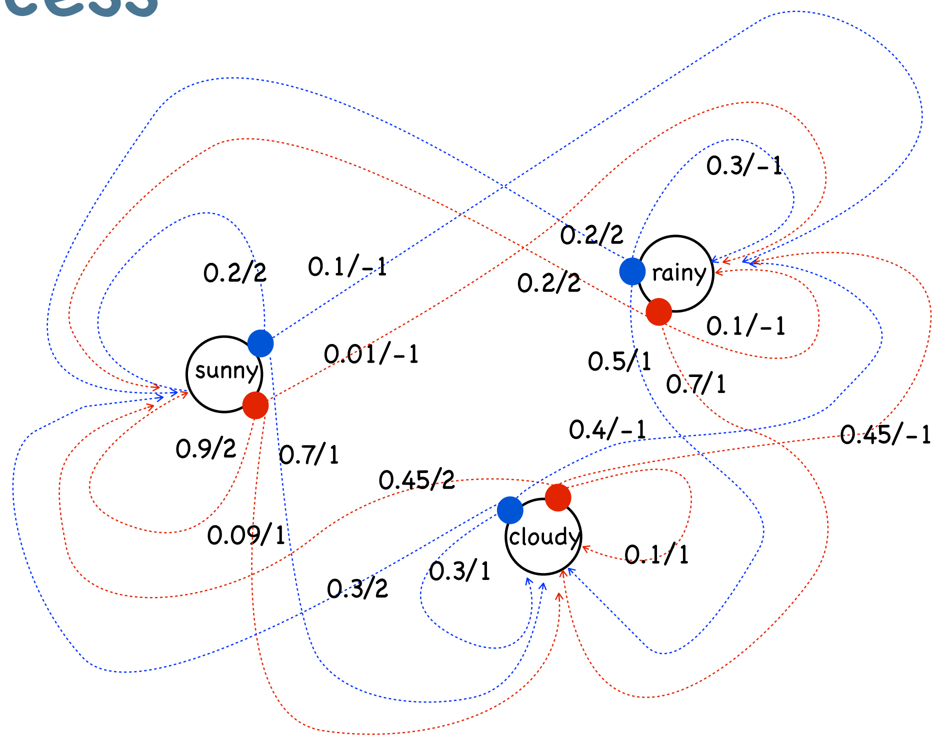
$\pi =$

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:

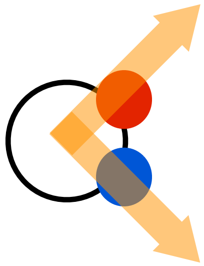
$$V^{\pi}(s) = \sum_{a} \pi(a|s) \sum_{s'} P(s'|s,a)\big(R(s,a,s') + V^{\pi}(s')\big)$$

expectation over actions

with respect to the policy

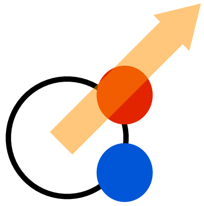
Q-function

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:

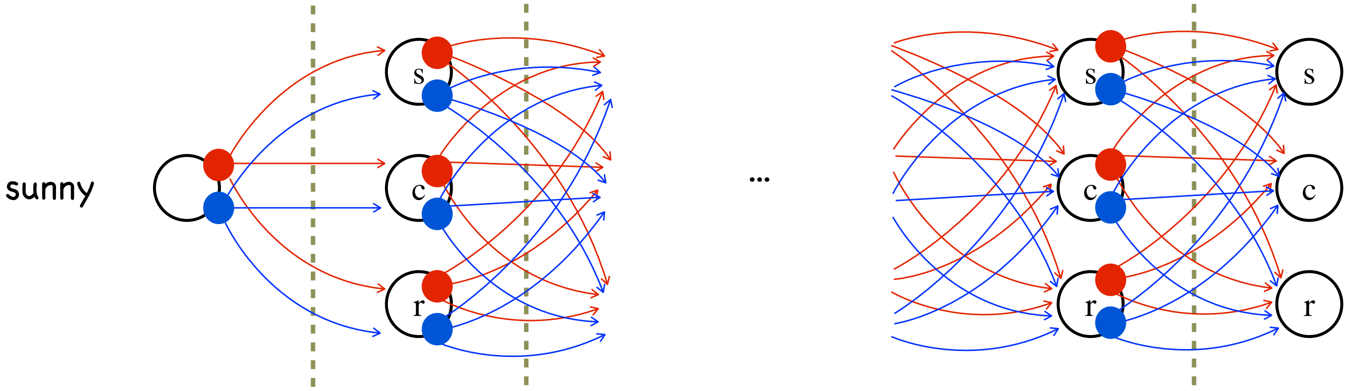
```
$V_0 = 0$
for $t$=0, 1, ...
    for all $s$    <- synchronous v.s. asynchronous
        $V_{t+1}(s) = \max_a \sum_{s'} P(s'|s,a)\big(R(s,a,s') + \gamma V_t(s)\big)$
    end for
    break if $|| V_{t+1} - V_t ||_\infty$ is small enough
end for
```

recall the optimal value function about V

Solve optimal policy in MDP

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

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



Dynamic programming

R. E. Bellman

1920-1984

Complexity

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

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

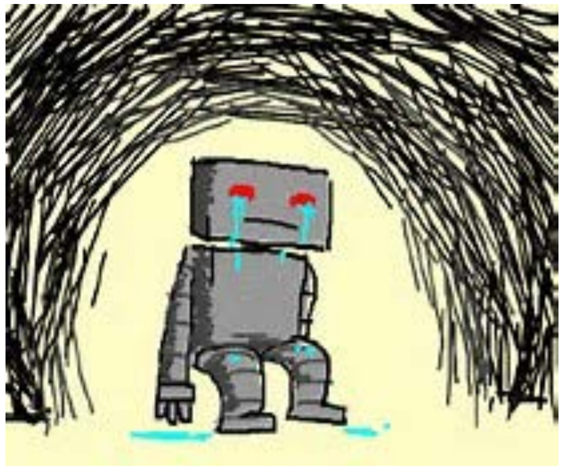
curse of dimensionality: Go board 19x19, $|S|=2.08\times 10^{170}$

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

from MDP to reinforcement learning

MDP $<S, A, R, P>$

R and P are unknown



Methods

A: learn R and P,

then solve the MDP

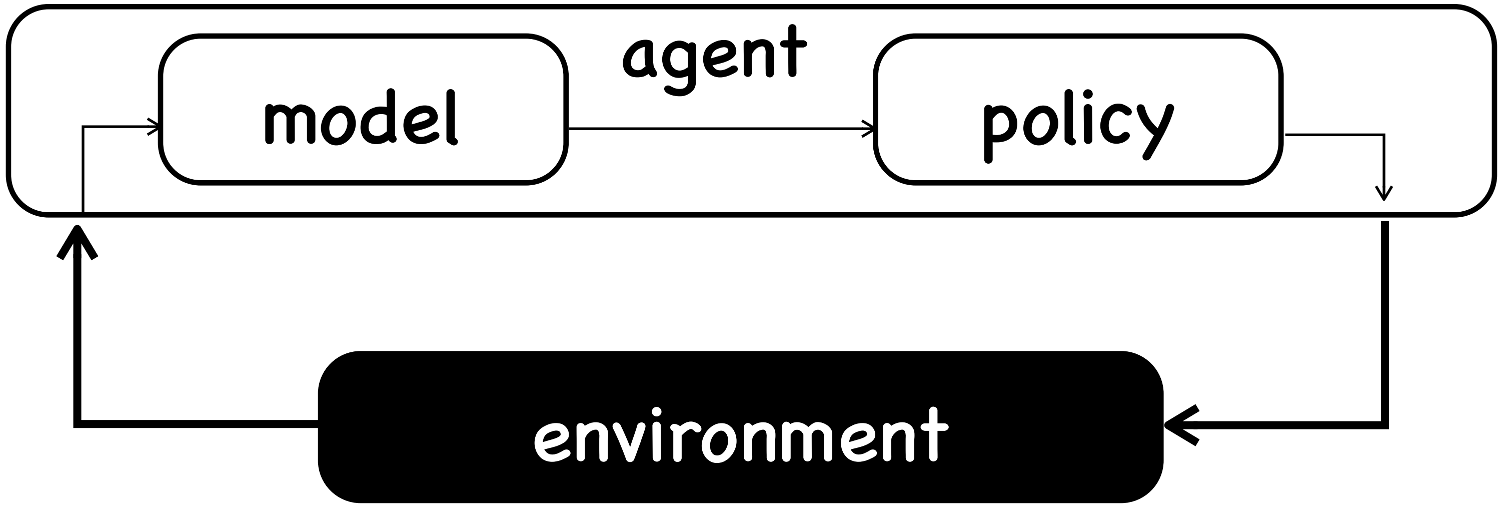
model-based

B: learn policy without R or P

model-free

MDP is the model

Model-based RL



basic idea:

1. explore the environment randomly,
2. build the model from observations,
3. find the policy by VI or PI

issues:

how to learn the model efficiently?

how to update the policy efficiently?

how to combine model learning and policy learning?

...

learn an MDP model

random walk, and record the transition and the reward.

more efficiently, visit unexplored states

RMax algorithm:

[Bertsekas, Tsitsiklis. R-Max---A general polynomial time algorithm for near-optimal reinforcement learning. JMLR’02]

```
initialize $R(s)$=$R$max, P = self-trainsition
loop
    choose action $a$, observe state $s$’ and reward $r$
    update transition count and reward count for $s,a,s$’
    if count of $s,a >= m$
        update reward and transition from estimations
    s = s’
```

sample complexity $\tilde{O}(|S|^2|A|V_{\max}^3/(\epsilon(1-\gamma))^3)$

[Strehl, et al. Reinforcement learning in finite MDPs: PAC analysis. JMLR’09]

Model-free RL

explore the environment and learn policy at the same time

Monte-Carlo method

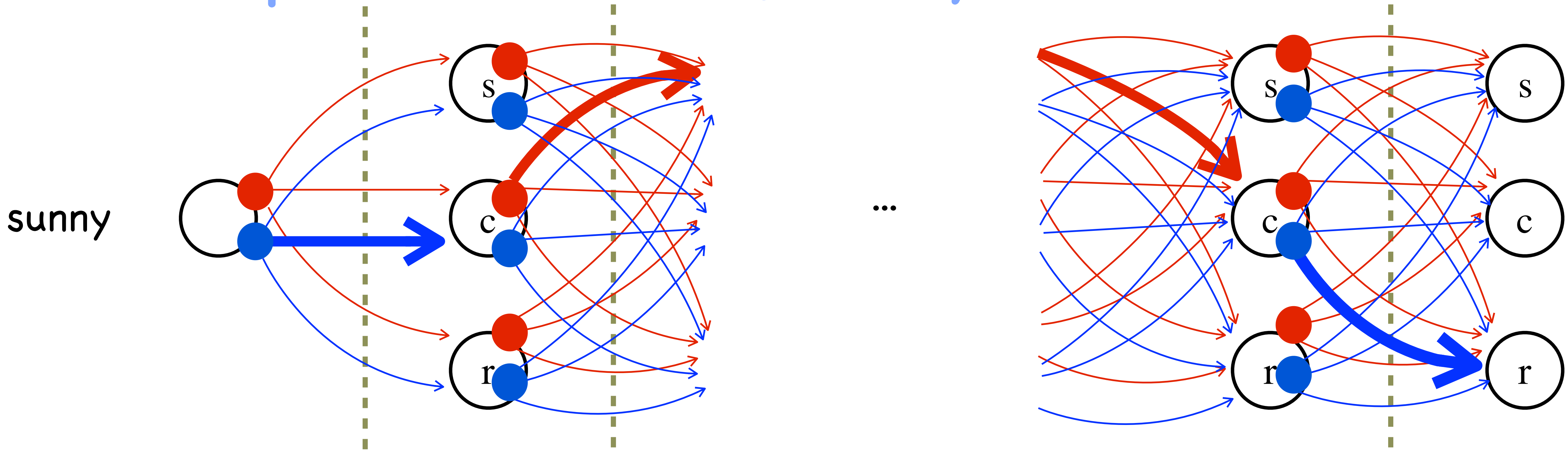
Temporal difference method

Monte Carlo RL - evaluation

Q, not V

expected total reward $Q^\pi(s,a) = E[\sum_{t=1}^{T} r_t | s, a]$

expectation of trajectory-wise rewards



sample trajectory m times,

approximate the expectation by average

$$Q^\pi(s,a) = \frac{1}{m}\sum_{i=1}^{m} R(\tau_i)$$

τ_i is sample by following π after s,a

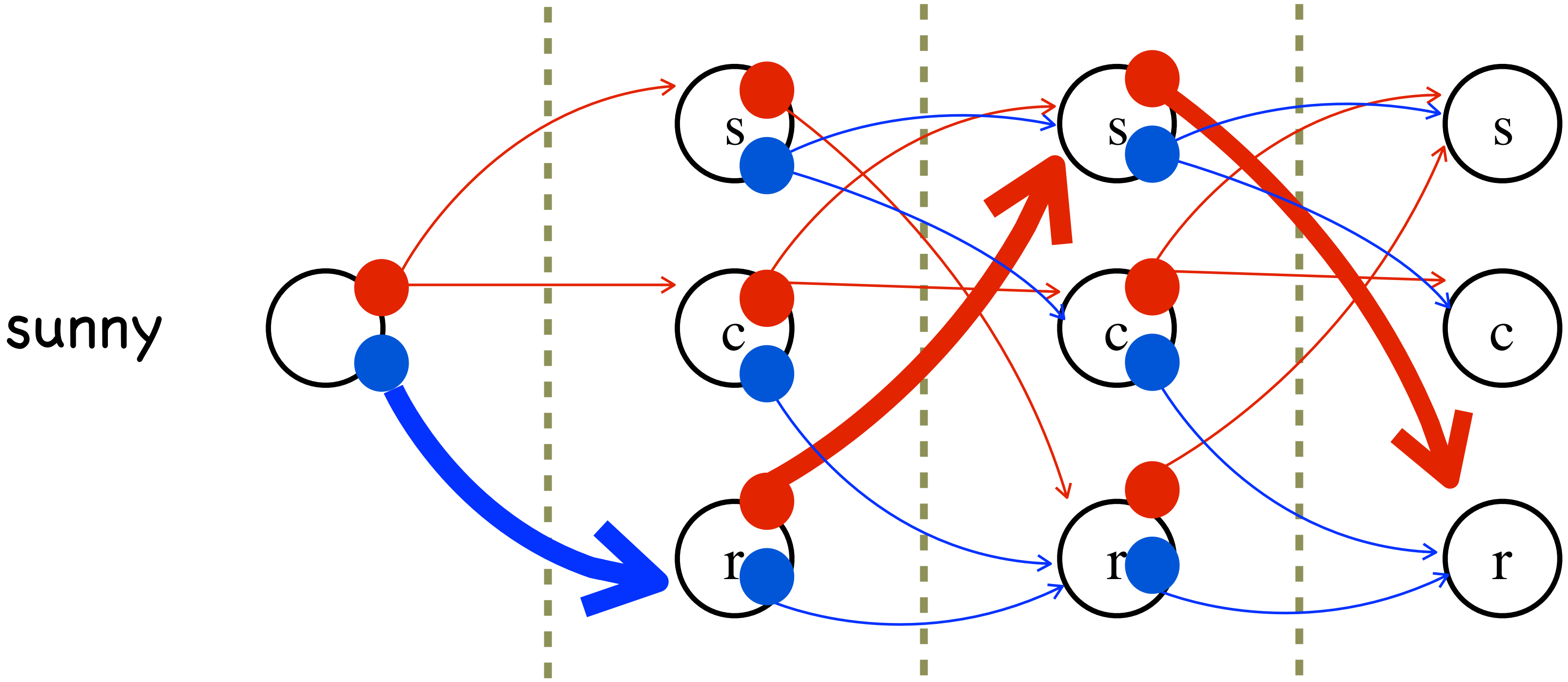
Monte Carlo RL - evaluation+improvement

```
$Q_0 = 0$
for $i$=0, 1, ..., m
    generate trajectory $<s_0, a_0, r_1, s_1, ..., s_T>$
    for $t$=0, 1, ..., $T$-1
        R = sum of rewards from $t$ to $T$
        $Q(s_t,a_t) = (\mathrm{c}(s_t,a_t)\, Q(s_t,a_t) + \mathrm{R}) / (\mathrm{c}(s_t,a_t) + 1)$
        $\mathrm{c}(s_t,a_t)$++
    end for
    update policy $\pi(s) = \arg\max_a Q(s,a)$
end for
```

improvement ?

Monte Carlo RL

problem: what if the policy takes only one path?



cannot improve the policy

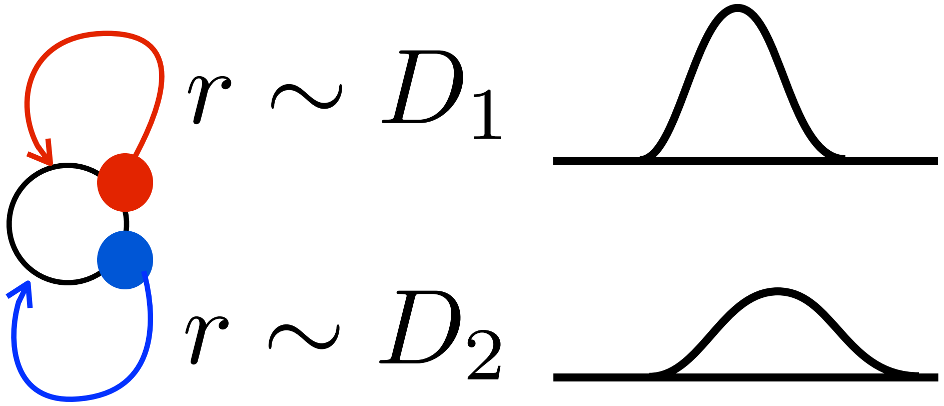
no exploration of the environment

needs exploration !

Exploration methods

one state MDP:

a.k.a. bandit model



maximize the long-term total reward

- exploration only policy: try every action in turn

  waste many trials

- exploitation only policy: try each action once, follow the best action forever

  risk of pick a bad action

balance between exploration and exploitation

Exploration methods

ϵ-greedy:

follow the best action with probability 1-ϵ

choose action randomly with probability ϵ

ϵ should decrease along time

softmax:

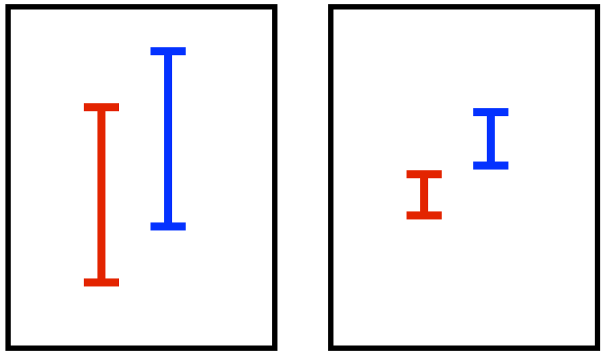
probability according to action quality

$$P(k) = e^{Q(k)/\theta} / \sum_{i=1}^{K} e^{Q(i)/\theta}$$

upper confidence bound (UCB):

choose by action quality + confidence

$$Q(k) + \sqrt{2 \ln n / n_k}$$



Action-level exploration

ϵ-greedy policy:

given a policy π

$$\pi_\epsilon(s) = \begin{cases} \pi(s), \text{with prob. } 1-\epsilon \\ \text{randomly chosen action, with prob. } \epsilon \end{cases}$$

ensure probability of visiting every state > 0

exploration can also be in other levels

Monte Carlo RL

```
$Q_0 = 0$
for $i$=0, 1, ..., m
    generate trajectory $<s_0, a_0, r_1, s_1, ..., s_T>$ by $\pi_\epsilon$
    for $t$=0, 1, ..., $T$-1
        R = sum of rewards from $t$ to $T$
        $Q(s_t, a_t) = (c(s_t, a_t) Q(s_t, a_t) + R) / (c(s_t, a_t) + 1)$
        $c(s_t, a_t)$++
    end for
    update policy $\pi(s) = \arg\max_a Q(s, a)$
end for
```

Monte Carlo RL - on/off-policy

this algorithm evaluates π_ϵ !   on-policy

what if we want to evaluate π ?   off-policy

importance sampling:

$$E[f] = \int_x p(x) f(x) \mathrm{d}x = \int_x q(x) \frac{p(x)}{q(x)} f(x) \mathrm{d}x$$

sample from p sample from q

$$\frac{1}{m} \sum_{i=1}^{m} f(x) \qquad\qquad \frac{1}{m} \sum_{i=1}^{m} \frac{p(x)}{q(x)} f(x)$$

Monte Carlo RL -- off-policy

```
$Q_0 = 0$
for $i$=0, 1, ..., m
    generate trajectory $<s_0, a_0, r_1, s_1, ..., s_T>$ by $\pi_\epsilon$
    for $t$=0, 1, ..., $T$-1
        R = sum of rewards from $t$ to $T \times \prod_{i=t+1}^{T-1} \frac{\pi(x_i, a_i)}{p_i}$
        $Q(s_t, a_t) = (\mathrm{c}(s_t, a_t) Q(s_t, a_t) + \mathrm{R}) / (\mathrm{c}(s_t, a_t) + 1)$
        $\mathrm{c}(s_t, a_t)$++
    end for
    update policy $\pi(s) = \arg\max_a Q(s, a)$
end for
```

$$p_i = \begin{cases} 1 - \epsilon + \epsilon/|A|, a_i = \pi(s_i), \\ \epsilon/|A|, a_i \neq \pi(s_i) \end{cases}$$

Monte Carlo RL

summary

Monte Carlo evaluation:

approximate expectation by sample average

action-level exploration

on-policy, off-policy: importance sampling

Monte Carlo RL:

evaluation + action-level exploration + policy improvement (on/off-policy)

Incremental mean

$$Q(s_t,a_t)= (\mathrm{c}(s_t,a_t)\,Q(s_t,a_t)+\mathrm{R})/(\mathrm{c}(s_t,a_t)+1)$$

$$\mu_t = \frac{1}{t}\sum_{i=1}^{t} x_i = \frac{1}{t}(x_t + \sum_{i=1}^{t-1} x_i) = \frac{1}{t}(x_t + (t-1)\mu_{t-1})$$

$$= \mu_{t-1} + \frac{1}{t}(x_t - \mu_{t-1})$$

In general, $\mu_t = \mu_{t-1} + \alpha(x_t - \mu_{t-1})$

Monte-Carlo update:

$$Q(s_t,a_t) \Leftarrow Q(s_t,a_t) + \alpha\underbrace{(R - Q(s_t,a_t))}_{\text{MC error}}$$

Temporal-Difference Learning - evaluation

update policy online learn as you go

TD Evaluation

Monte-Carlo update:

$$Q(s_t, a_t) \Leftarrow Q(s_t, a_t) + \alpha\underbrace{(R - Q(s_t, a_t))}_{\text{MC error}}$$

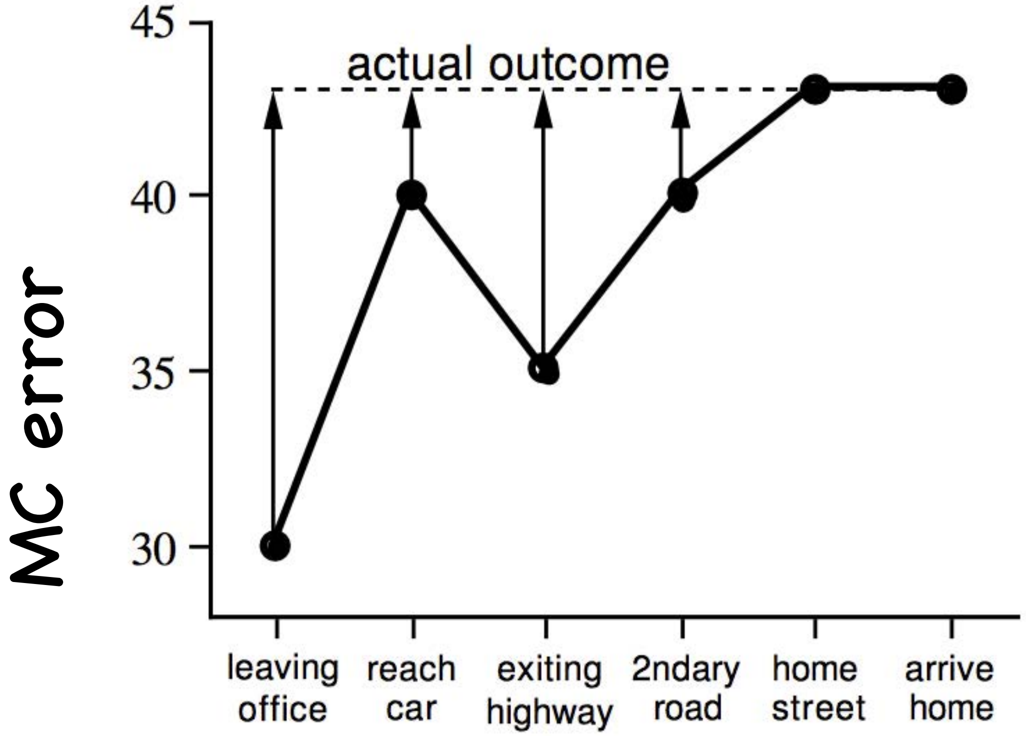
TD update:

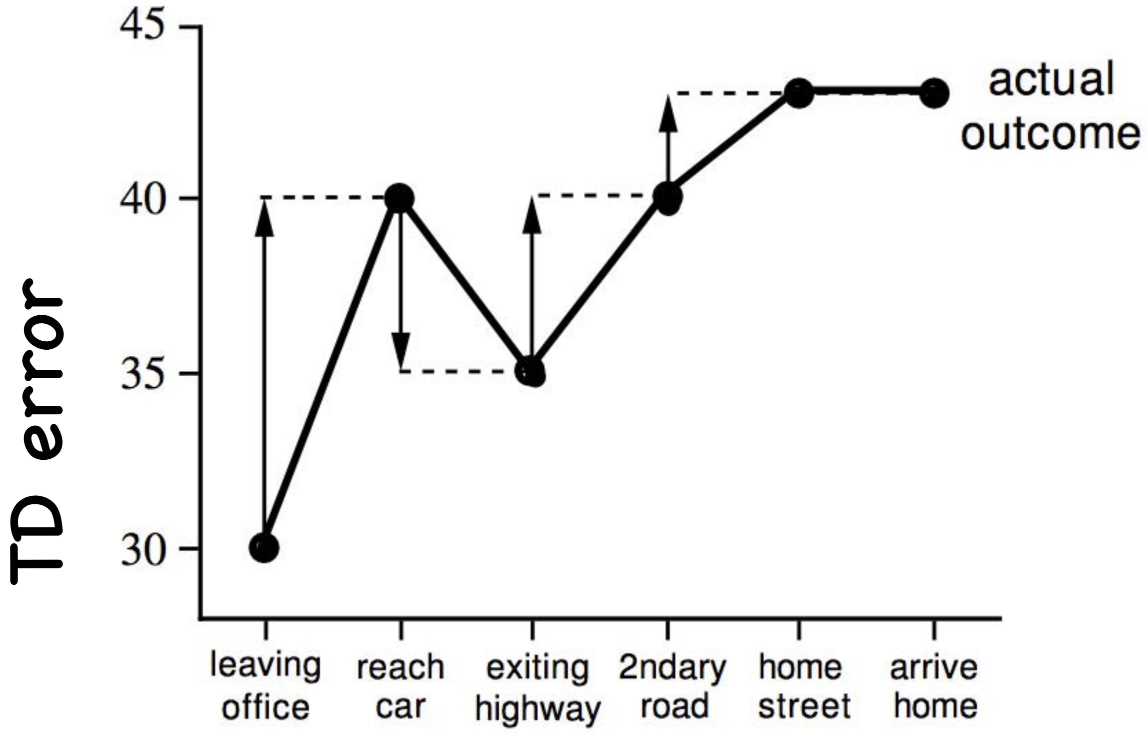
$$Q(s_t, a_t)$$

$$\Leftarrow Q(s_t, a_t) + \alpha\underbrace{(r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))}_{\text{TD error}}$$

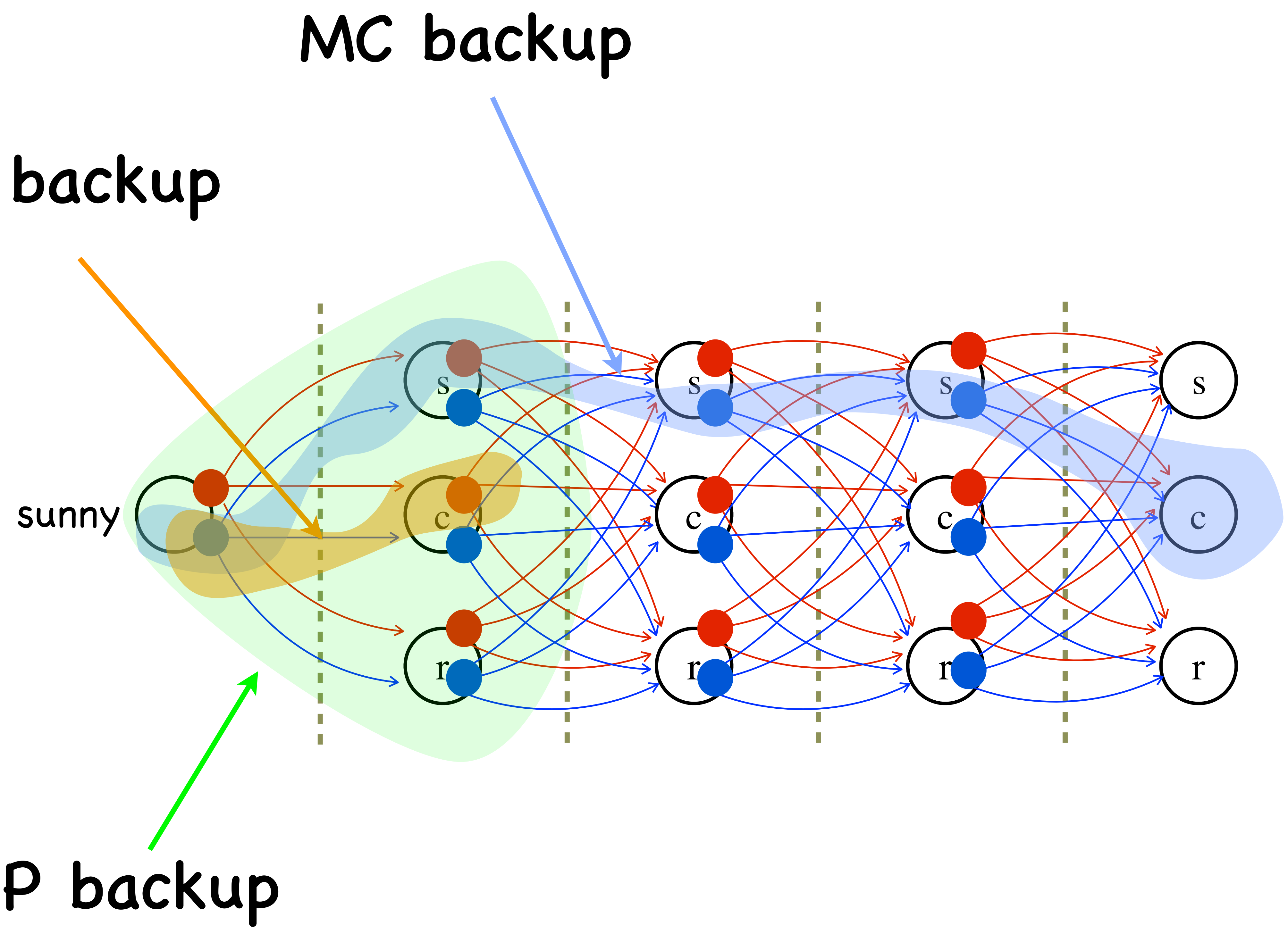
Temporal-Difference Learning - example

state	elapsed time	predicted remaining time	predicted total time
leaving office	0	30	30
reach car, raining	5	35	40
exit highway	20	15	35
behind truck	30	10	40
home street	40	3	43
arrive home	43	0	43





Temporal-Difference Learning - backups



MC backup

TD backup

DP backup

SARSA

On-policy TD control

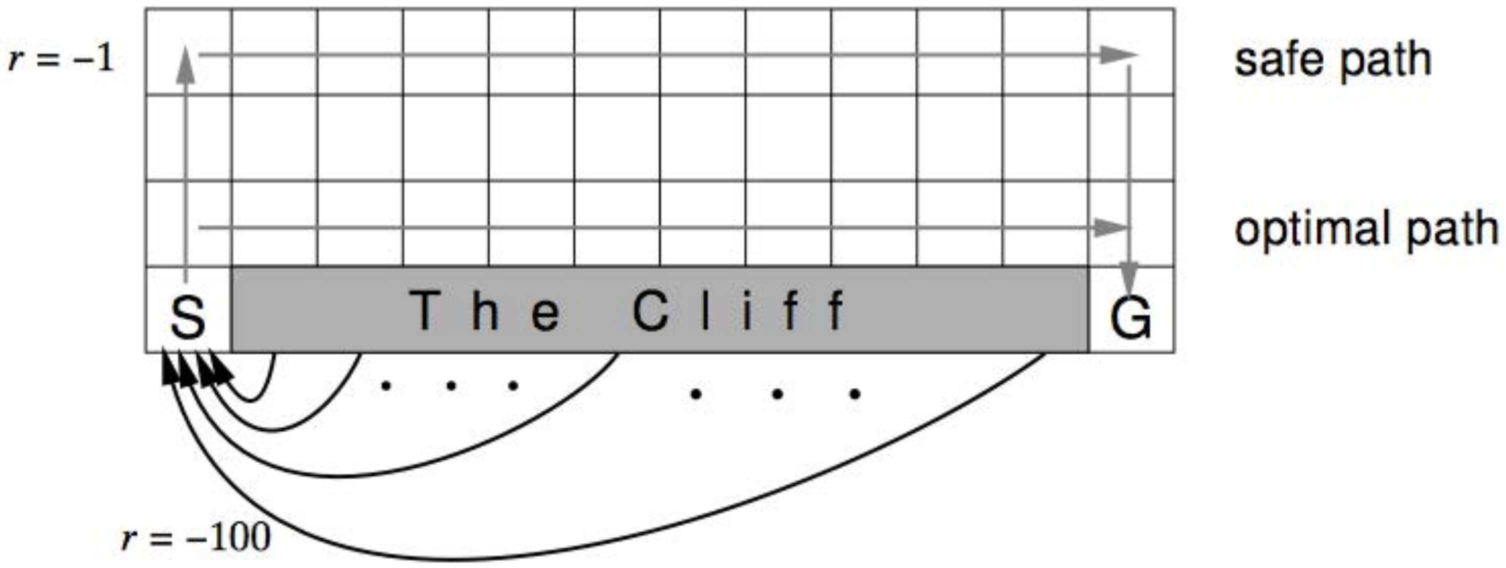
```
$Q_0 = 0$, initial state
for $i$=0, 1, ...
    $a = \pi_\epsilon(s)$
    $s$’, $r$ = do action $a$
    $a$’ $= \pi_\epsilon(s')$
    $Q(s,a) \mathrel{+}= \alpha(r + \gamma Q(s',a') - Q(s,a))$
    $\pi(s) = \arg\max_a Q(s,a)$
    $s = s$’
end for
```

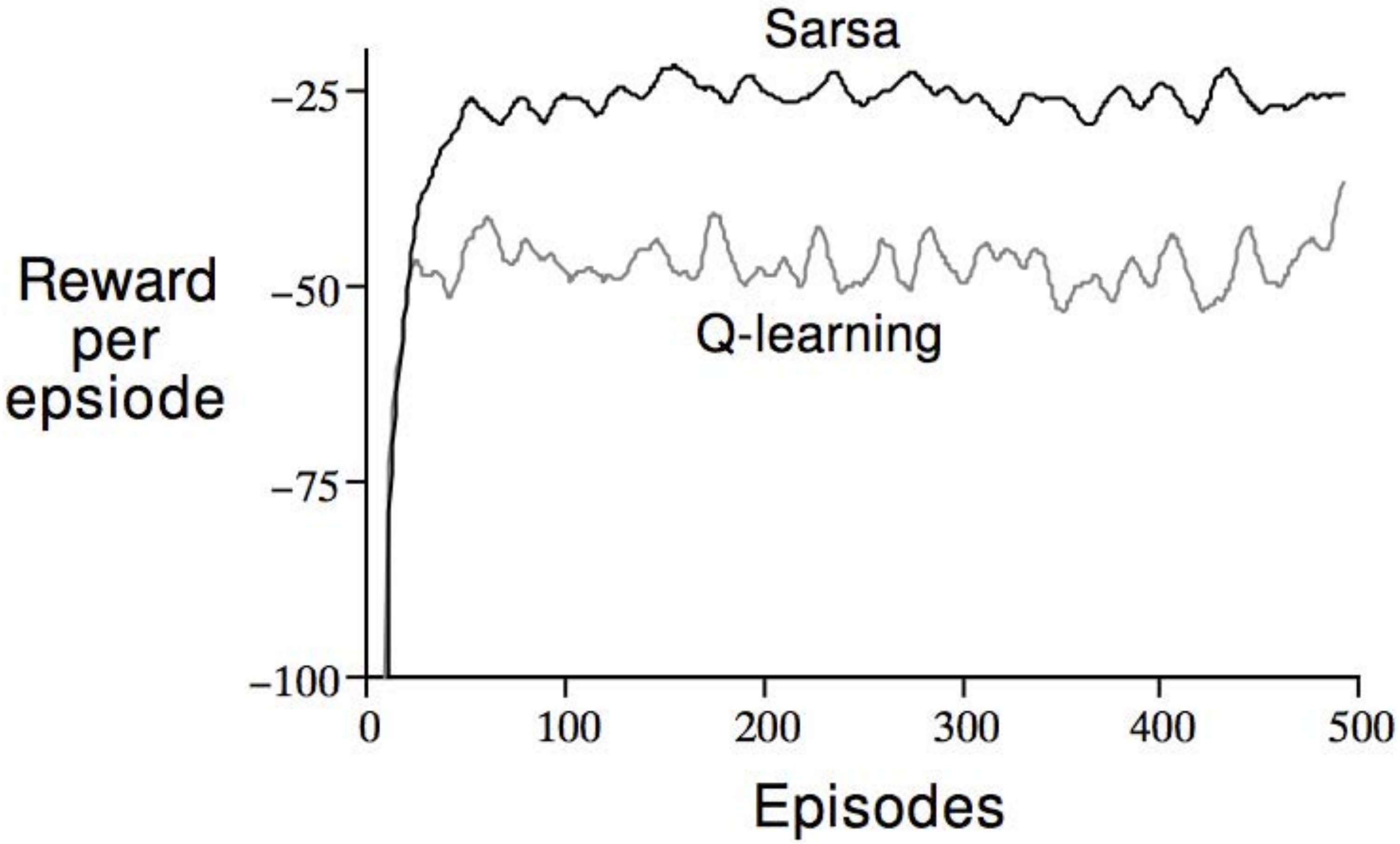
Q-learning

Off-policy TD control

```
$Q_0 = 0$, initial state
for $i$=0, 1, ...
    $a = \pi_\epsilon(s)$
    $s', r =$ do action $a$
    $a' = \pi(s')$
    $Q(s,a) \mathrel{+}= \alpha(r + \gamma Q(s',a') - Q(s,a))$
    $\pi(s) = \arg\max_a Q(s,a)$
    $s = s'$
end for
```

SARSA v.s. Q-learning



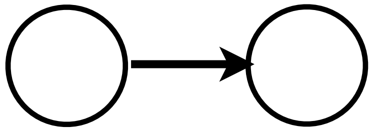


λ-return

in between TD and MC: n-step prediction

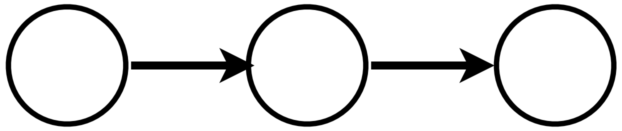
n-step return

TD(1-step)



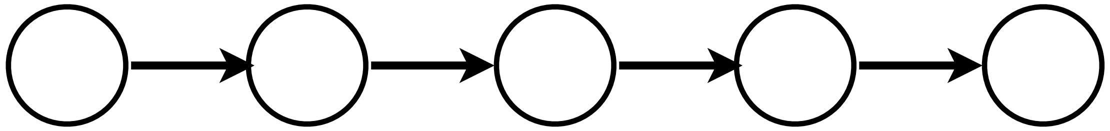
$$R^{(1)} = r_{t+1} + \gamma Q(s_{t+1}, a_{t+1})$$

TD(2-step)



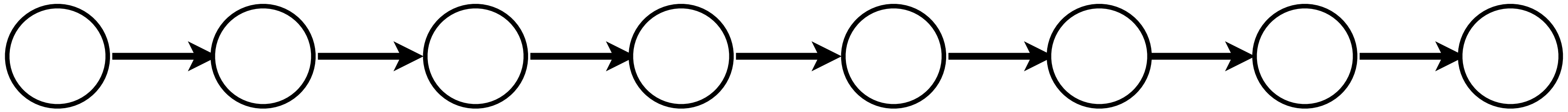
$$R^{(2)} = r_{t+1} + \gamma r_{t+2} + \gamma^2 Q(s_{t+2}, a_{t+2})$$

TD(n-step)



$$R^{(n)} = \sum_{i=1}^{n} \gamma^{i-1} r_{t+i} + \gamma^n Q(s_{t+n}, a_{t+n})$$

MC



$$R^{(\max)} = \sum_{i=1}^{T} \gamma^{i-1} r_{t+i}$$

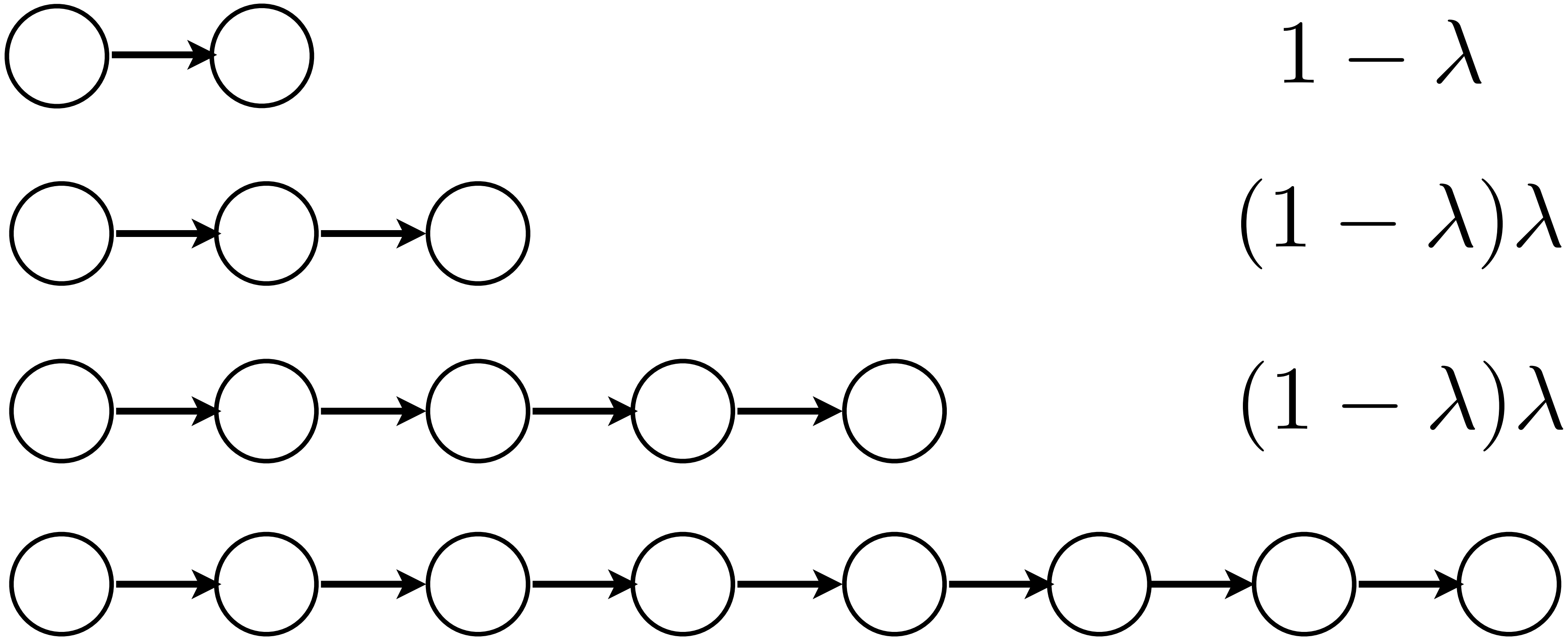
k-step TD:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha(R^{(k)} - Q(s_t, a_t))$$

λ-return

averaging k-step returns, parameter λ

weight



TD(1-step) $1-\lambda$

TD(2-step) $(1-\lambda)\lambda$

TD(n-step) $(1-\lambda)\lambda^{n-1}$

MC $(1-\lambda)\lambda^{\max -1}$

λ-return: $$R^\lambda = (1-\lambda)\sum_{k=1}^{\infty}\lambda^{k-1}R^k$$

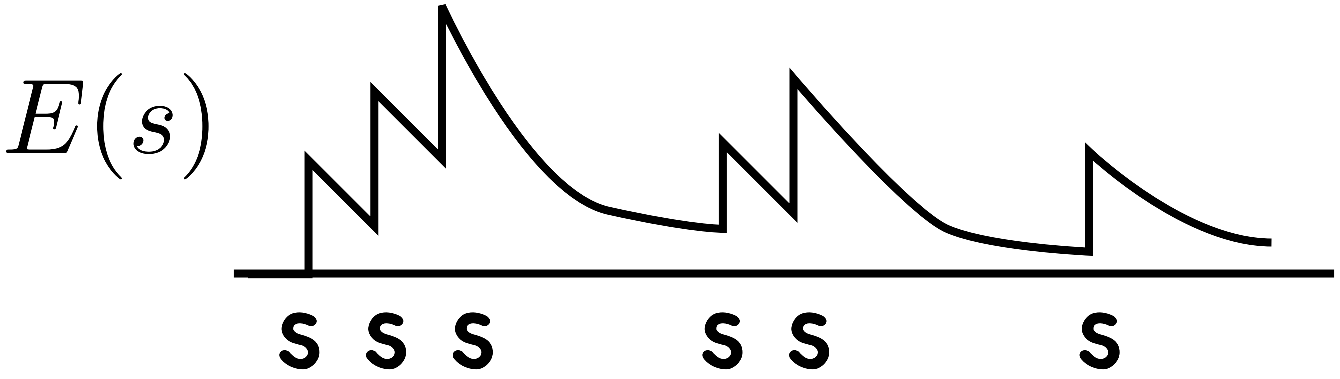
TD(λ): $$Q(s_t,a_t) = Q(s_t,a_t) + \alpha(R^\lambda - Q(s_t,a_t))$$

Implementation: eligibility traces

Maintain an extra memory E(s)

$$E_0(s,a) = 0$$

$$E_t(s,a) = \gamma\lambda E_{t-1}(s,a) + I(s_t = s, a_t = a)$$



TD(λ)

TD error:

$$\delta_t = r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)$$

Update:

$$Q(s,a) \Leftarrow Q(s,a) + \alpha\delta_t E_t(s,a)$$

SARSA(λ)

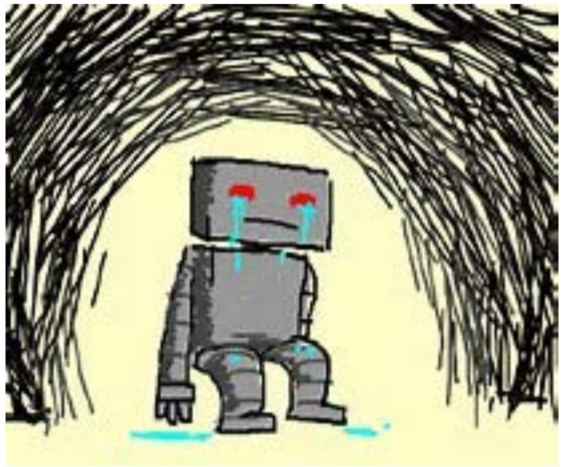
```
$Q_0 = 0$, initial state
for $i$=0, 1, ...
    $s$', $r$ = do action from policy $\pi_\epsilon$
    $a$' $= \pi_\epsilon(s')$
    $\delta = r + \gamma Q(s', a') - Q(s, a)$
    $E(s, a) + +$
    for all $s$, $a$
        $Q(s, a) = Q(s, a) + \alpha\delta E_t(s, a)$
        $E(s, a) = \gamma E(s, a)$
    end for
    $s$ = $s$', $a$ = $a$', $\pi(s) = \arg\max_a Q(s, a)$
end for
```

we can do RL now! … in (small) discrete state space

RL in continuous state space

MDP $<S, A, R, P>$

S (and A) is in $\mathbb{R}^n$



Value function approximation

modern RL

tabular representation

$\pi =$

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

very powerful representation

can be all possible policies !

linear function approx.

$$\hat{V}(s) = w^\top \phi(s)$$
$$\hat{Q}(s,a) = w^\top \phi(s,a)$$
$$\hat{Q}(s,a_i) = w_i^\top \phi(s)$$

ϕ is a feature mapping

w is the parameter vector

may not represent all policies !

Value function approximation

to approximate Q and V value function

least square approximation

$$J(w) = E_{s\sim\pi}[\big(Q^\pi(s,a) - \hat{Q}(s,a)\big)^2]$$

online environment: stochastic gradient on single sample

$$\Delta w_t = \theta(Q^\pi(s_t,a_t) - \hat{Q}(s_t,a_t))\nabla_w \hat{Q}(s_t,a_t)$$

replace

Recall the errors:

MC update: $Q(s_t,a_t) += \alpha(\underline{R} - \underline{Q(s_t,a_t)})$

TD update: $Q(s_t,a_t) += \alpha(\underline{r_{t+1} + \gamma Q(s_{t+1},a_{t+1})} - \underline{Q(s_t,a_t)})$

target

model

Value function approximation

MC update:

$$\Delta w_t = \theta(R - \hat{Q}(s_t, a_t))\nabla_w \hat{Q}(s_t, a_t)$$

TD update:

$$\Delta w_t = \theta(r_{t+1} + \gamma\hat{Q}(s_{t+1}, a_{t+1}) - \hat{Q}(s_t, a_t))\nabla_w \hat{Q}(s_t, a_t)$$

eligibility traces

$$E_t = \gamma\lambda E_{t-1} + \nabla_w \hat{Q}(s_t, a_t)$$

Q-learning with function approximation

```
$w = 0$, initial state
for $i$=0, 1, ...
    $a = \pi_\epsilon(s)$
    $s$’, $r$ = do action $a$
    $a$’ = $\pi(s')$
    $w+ = \theta(r + \gamma\hat{Q}(s,a) - \hat{Q}(s,a))\nabla_w\hat{Q}(s_t,a_t)$
    $\pi(s) = \arg\max_a \hat{Q}(s,a)$
    $s$ = $s$’
end for
```

Approximation model

Linear approximation $\hat{Q}(s,a) = w^\top \phi(s,a)$

$$\nabla_w \hat{Q}(s,a) = \phi(s,a)$$

coarse coding: raw features

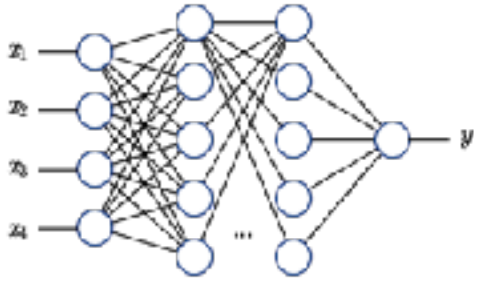
discretization: tide with indicator features

kernelization:

$$\hat{Q}(s,a) = \sum_{i=1}^{m} w_i K((s,a),(s_i,a_i))$$

(s_i, a_i) can be randomly sampled

Approximation model



Nonlinear model approximation $\hat{Q}(s,a) = f(s,a)$

neural network: differentiable model

recall the TD update:

$$\Delta w_t = \theta(r_{t+1} + \gamma \hat{Q}(s_{t+1}, a_{t+1}) - \hat{Q}(s_t, a_t))\underline{\nabla_w \hat{Q}(s_t, a_t)}$$

follow the BP rule to pass the gradient

Batch RL methods

gradient on single sample introduces large variance

Batch mode evaluation:

collect trajectory and history data

$$D = \{(s_1, V_1^\pi), (s_2, V_2^\pi), \ldots, (s_m, V_m^\pi)\}$$

solve batch least square objective

$$J(w) = E_D[\left(V^\pi - \hat{V}(s)\right)^2]$$

linear function: closed form

neural networks: batch update/repeated stochastic update

LSMC, LSTD, LSTD(λ)

Batch RL methods

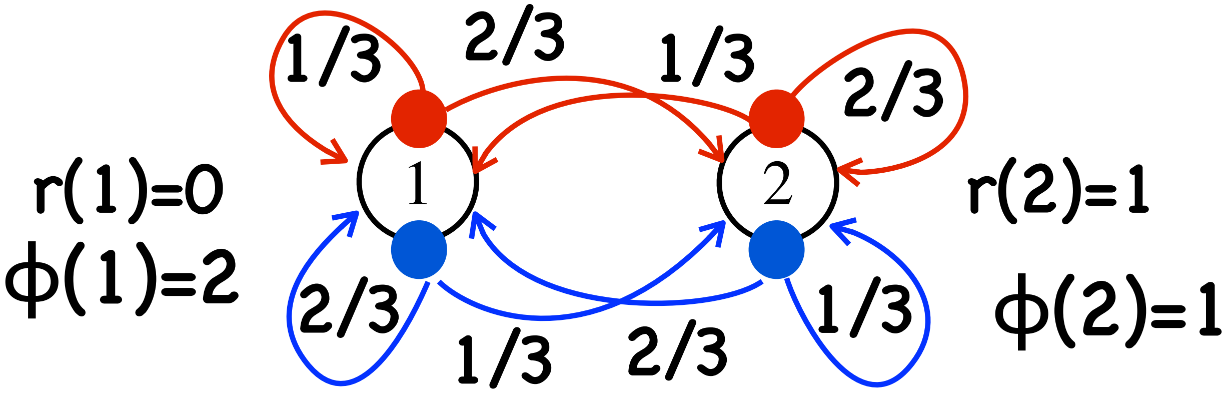
gradient on single sample introduces large variance

Batch mode policy iteration: LSPI

```
$Q_0 = 0$, initial state
for $i$=0, 1, ...
    collect data $D$
    $w = \arg\min_w \sum_{(s,a)\in D} (r + \gamma\hat{Q}(s,\pi(s)) - \hat{Q}(s,a))\phi(s,a)$
    $\forall s, \pi(s) = \arg\max_a Q(s,a)$
end for
```

policy degradation in value function based methods

[Bartlett. An Introduction to Reinforcement Learning Theory: Value Function Methods. Advanced Lectures on Machine Learning, LNAI 2600]



optimal policy: red

V*(2) > V*(1) > 0

let $\hat{V}(s) = w\phi(s)$, to ensure $\hat{V}(2) > \hat{V}(1)$, $w < 0$

as value function based method minimizes $\|\hat{V} - V^*\|$
results in $w > 0$

sub-optimal policy, better value ≠ better policy

Policy Search

Parameterized policy

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

Gibbs policy (logistic regression)

$$\pi_\theta(i|s) = \frac{\exp(\theta_i^\top \phi(s))}{\sum_j \exp(\theta_j^\top \phi(s))}$$

Gaussian policy (continuous !)

$$\pi_\theta(a|s) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(\theta^\top s - a)^2}{\sigma^2}\right)$$

Policy search v.s. value function based

Policy search advantages:

- effective in high-dimensional and continuous action space
- learn stochastic policies directly
- avoid policy degradation

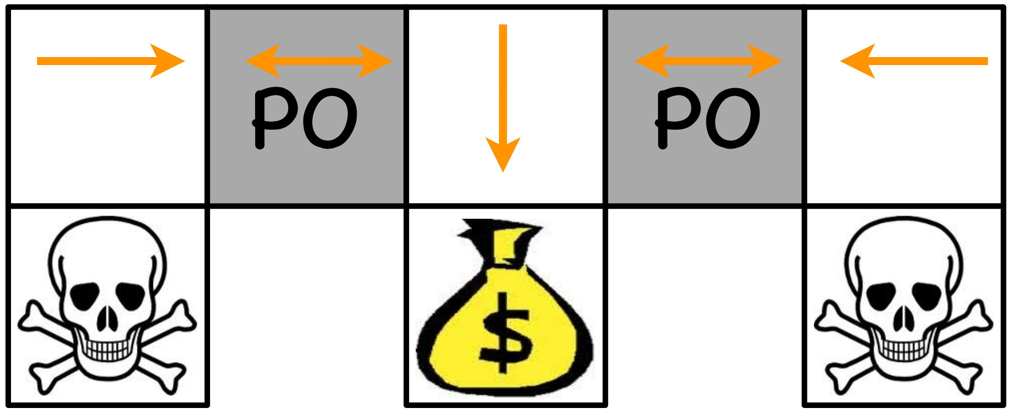
disadvantages:

- converge only to a local optimum
- high variance

Example: Aliased gridworld

state PO cannot be distinguished

=> same action distribution



deterministic policy: stuck at one side

value function based policy is mostly deterministic

stochastic policy: either direction with prob. 0.5

policy search derives stochastic policies

Direct objective functions

episodic environments: trajectory-wise total reward

$$J(\theta) = \int_{Tra} p_\theta(\tau) R(\tau)\ \mathrm{d}\tau$$

where $p_\theta(\tau) = p(s_0) \prod_{i=1}^{T} p(s_i | a_i, s_{i-1}) \pi_\theta(a_i | s_{i-1})$

is the probability of generating the trajectory

continuing environments: one-step MDPs

$$J(\theta) = \int_S d^{\pi_\theta}(s) \int_A \pi_\theta(a|s) R(s,a)\ \mathrm{d}s\ \mathrm{d}a$$

d^{π_θ} is the stationary distribution of the process

Optimization by sampling

finite difference

$$\frac{\partial J(\theta)}{\partial \theta} \approx \frac{J(\theta + \epsilon u_k) - J(\theta)}{\epsilon}$$

u_k is a dimension indicator, increase the parameter in one dimension a bit, evaluate the progress, choose the best dimension to proceed

simple, noisy, converges slowly

works for non-differentiable objectives

Analytical optimization: REINFORCE

$$J(\theta) = \int_{Tra} p_\theta(\tau) R(\tau)\ \mathrm{d}\tau$$

logarithm trick $\nabla_\theta p_\theta = p_\theta \nabla_\theta \log p_\theta$

as $p_\theta(\tau) = p(s_0) \prod_{i=1}^{T} p(s_i | a_i, s_{i-1}) \pi_\theta(a_i | s_{i-1})$

$$\nabla_\theta \log p_\theta(\tau) = \sum_{i=1}^{T} \nabla_\theta \log \pi_\theta(a_i | s_{i-1}) + \text{const}$$

gradient: $\nabla_\theta J(\theta) = \int_{Tra} p_\theta(\tau) \nabla_\theta \log p_\theta(\tau) R(\tau)\ \mathrm{d}\tau$

$$= E[\sum_{i=1}^{T} \nabla_\theta \log \pi_\theta(a_i | s_i) R(s_i, a_i)]$$

use samples to estimate the gradient (unbiased estimation)

Analytical optimization: REINFORCE

Gibbs policy $\pi_\theta(i|s) = \dfrac{\exp(\theta_i^\top \phi(s))}{\sum_j \exp(\theta_j^\top \phi(s))}$

$$\nabla_{\theta_j} \log \pi_\theta(a_i|s_i) = \begin{cases} \phi(s_i, a_i)(1 - \pi_\theta(a_i|s_i)), & i = j \\ -\phi(s_i, a_i)\pi_\theta(a_i|s_i) & i \neq j \end{cases}$$

Gaussian policy $\pi_\theta(a|s) = \dfrac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\dfrac{(\theta^\top \phi(s) - a)^2}{\sigma^2}\right)$

$$\nabla_{\theta_j} \log \pi_\theta(a_i|s_i) = -2\frac{(\theta^\top \phi(s) - a)\phi(s)}{\sigma^2} + \text{const}$$

Analytical optimization: One-step MDPs

$$J(\theta) = \int_S d^{\pi_\theta}(s) \int_A \pi_\theta(a|s) R(s,a) \ \mathrm{ds} \ \mathrm{da}$$

logarithm trick $\nabla_\theta \pi_\theta = \pi_\theta \nabla_\theta \log \pi_\theta$

$$\nabla_\theta J(\theta) = \int_S d^{\pi_\theta}(s) \int_A \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s) R(s,a) \ \mathrm{ds} \ \mathrm{da}$$
$$= E[\nabla_\theta \log \pi_\theta(a|s) R(s,a)]$$

equivalent to $E[\sum_{i=1}^{T} \nabla_\theta \log \pi_\theta(a_i|s_i) R(s_i, a_i)]$

use samples to estimate the gradient (unbiased estimation)

Reduce variance by critic: Actor-Critic

Maintain another parameter vector w

$$Q_w(s,a) = w^\top \phi(s,a) \approx Q^\pi(s,a)$$

value-based function approximated methods to update Q_w

MC, TD, TD(λ), LSPI

Multi-step MDPs: $J(\theta) = \int_S d^{\pi_\theta}(s) \int_A \pi_\theta(a|s) Q^{\pi_\theta}(s,a) \, \mathrm{d}s \, \mathrm{d}a$

$$\nabla_\theta J(\theta) = E[\nabla_\theta \log \pi_\theta(a|s) Q^{\pi_\theta}(s,a)]$$

Policy Gradient Theorem

equivalent gradient for all objectives

[Sutton et al. Policy gradient methods for reinforcement learning with function approximation. NIPS'00]

$$\nabla_\theta J(\theta) \approx E[\nabla_\theta \log \pi_\theta(a|s) Q_w(s,a)]$$

$$\text{if } w \text{ is a minimizer of } E[(Q^{\pi_\theta}(s,a) - Q_w(s,a))^2]$$

Learn policy (actor) and Q-value (critic) simultaneously

Example

```
initial state $s$
for $i$=0, 1, ...
    $a = \pi_\epsilon(s)$
    $s'$, $r$ = do action $a$
    $a' = \pi_\epsilon(s')$
    $\delta = r + \gamma Q_w(s', a') - Q_w(s, a)$
    $\theta = \theta + \nabla_\theta \log \pi_\theta(a|s) Q_w(s, a)$
    $w = w + \alpha\delta\phi(s, a)$
    $s = s'$, $a = a'$
end for
```

Control variance by introducing a bias term

for any bias term b(s)

$$\int_S d^{\pi_\theta}(s)\nabla_\theta \int_A \pi_\theta(a|s)\pi_\theta(a|s)b(s)\ \mathrm{dsda} = 0$$

gradient with a bias term

$$\nabla_\theta J(\theta) = E[\nabla_\theta \log \pi_\theta(a|s)(Q^\pi(s,a) - b(s))]$$

obtain the bias by minimizing variance

obtain the bias by V(s)

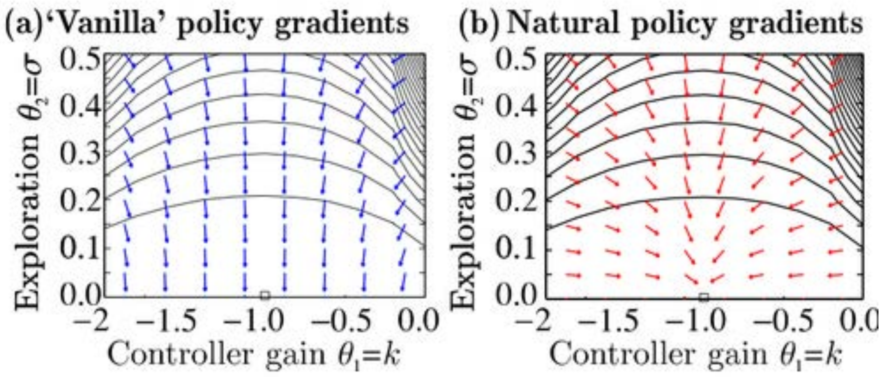
advantage function: $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$

$$\nabla_\theta J(\theta) = E[\nabla_\theta \log \pi_\theta(a|s) A^\pi(s,a)]$$

learn policy, Q and V simultaneously

Other gradients

nature policy gradient



[Kakade. A Natural Policy Gradient. NIPS'01]

functional policy gradient

$$\pi_\Psi(a|\boldsymbol{s}) = \frac{\exp(\Psi(\boldsymbol{s},a))}{\sum_{a'}\exp(\Psi(\boldsymbol{s},a'))}$$

$$\Psi_t = \sum_{i=1}^{t} h_t$$

[Yu et al. Boosting nonparametric policies. AAMAS'16]

parameter-level exploration

$$\theta \sim \mathcal{N}$$

[Sehnke et al. Parameter-exploring policy gradients. Neural Networks'10]

Derivative-free optimization

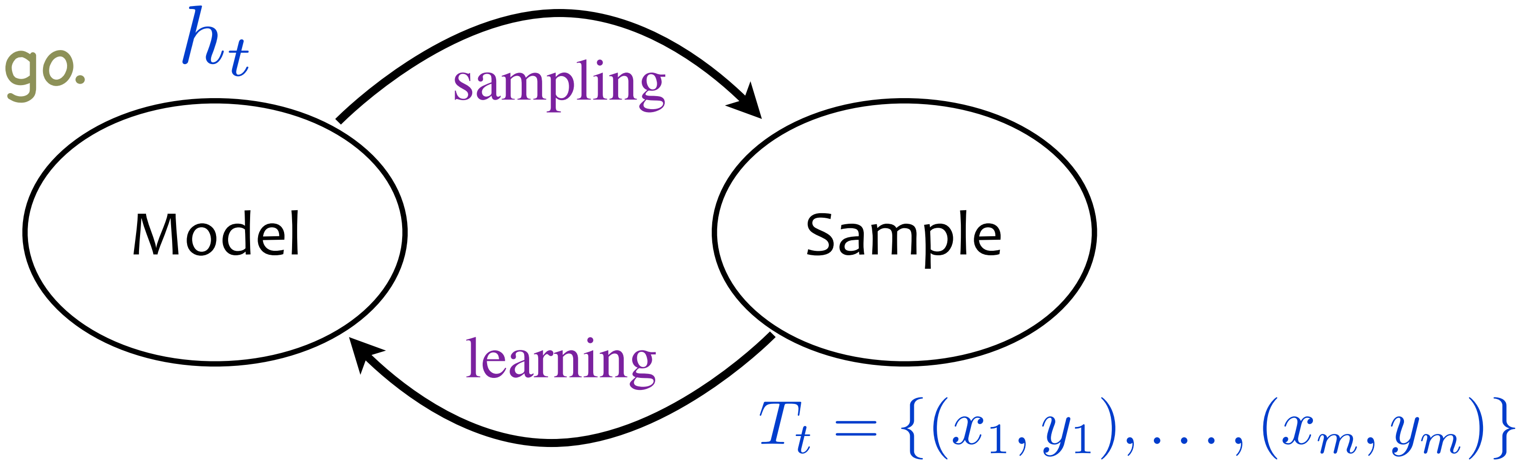
$$J(\theta) = \int_{Tra} p_\theta(\tau) R(\tau) \, \mathrm{d}\tau$$

For optimization problems $\arg\min_{x \in X} f(x)$

can only access the function value f(x) for optimization

Many derivative-free optimization methods are model-based

- CMA-ES
- Estimation of distribution algo.
- Cross-entropy
- ...

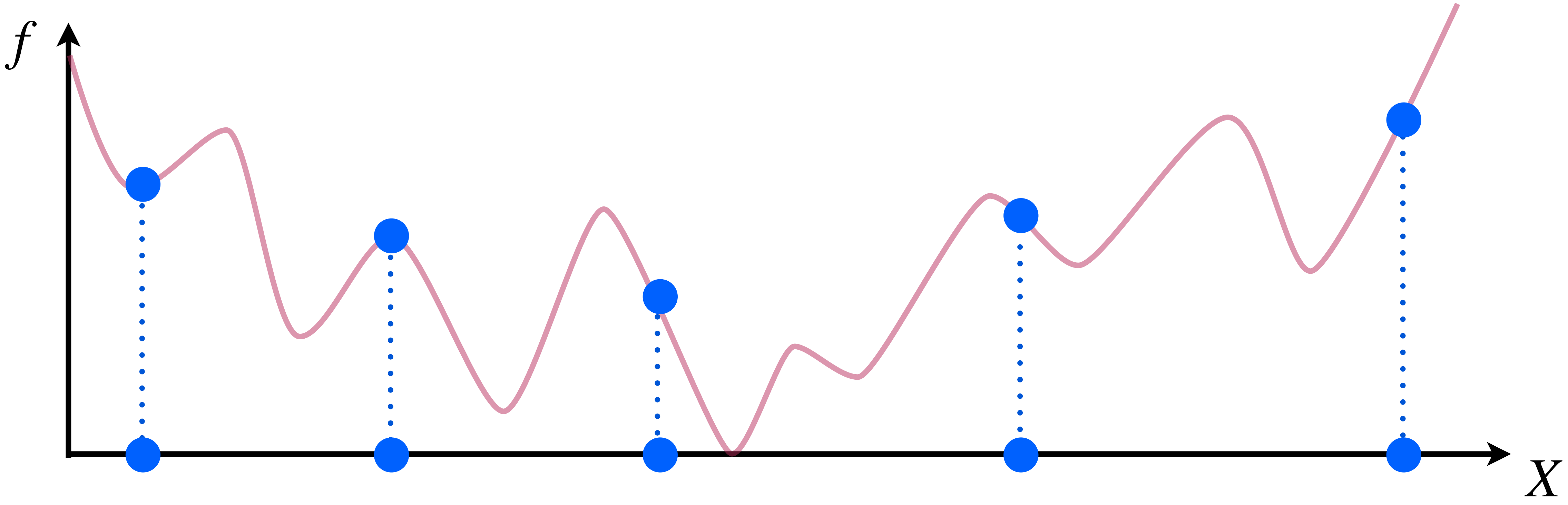


suitable for complex optimization problems

- not guided by gradient
- non-convex, many local optima, non-differentiable, non-continuous

Derivative-free optimization

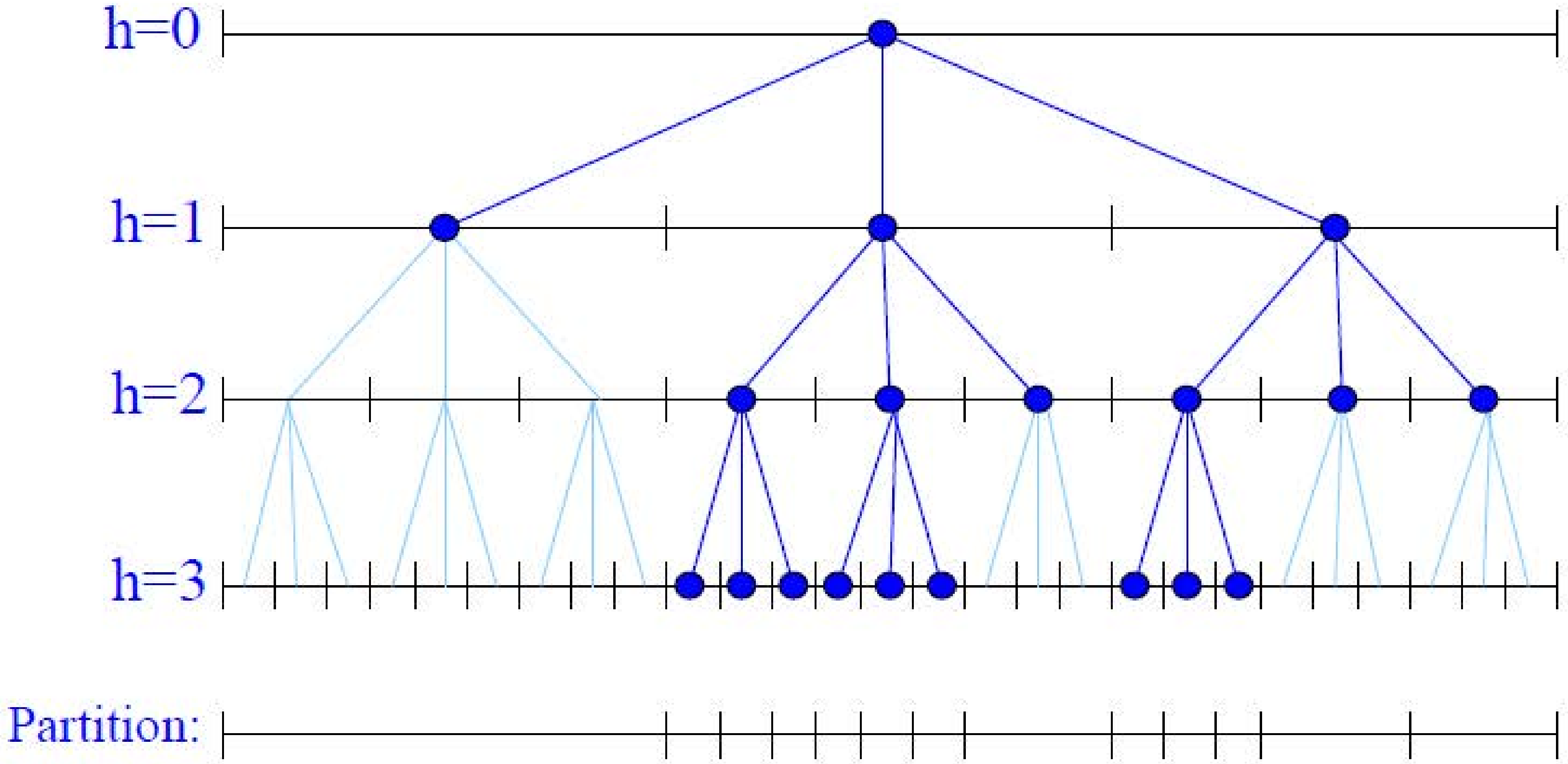
Intuition: sampling can disclose the optimization function

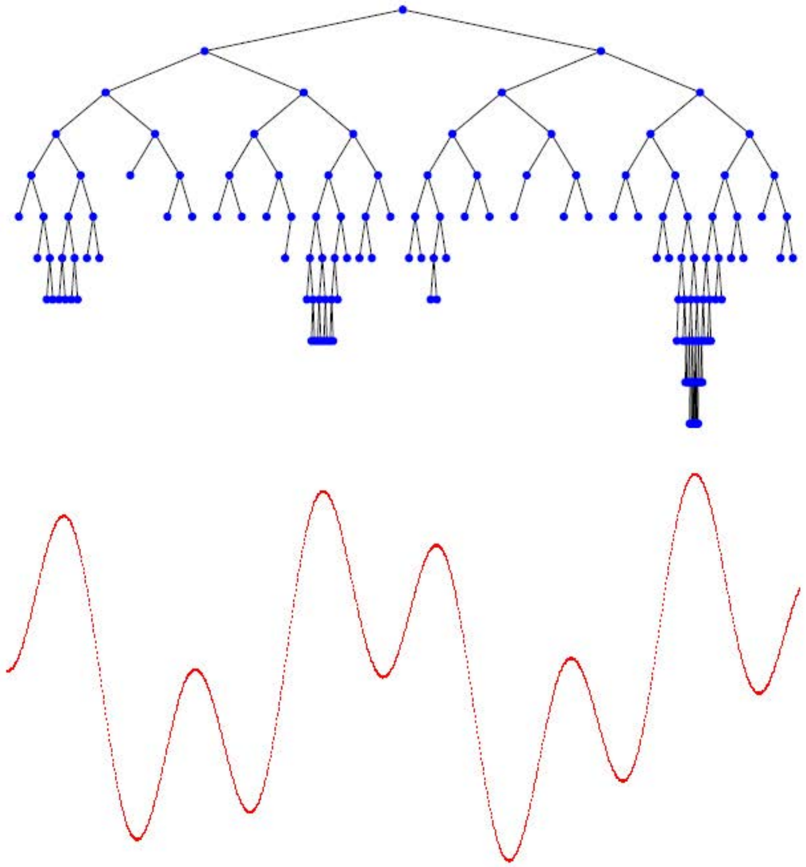


Recent development

- Optimistic optimization
- Bayesian optimization
- Classification-based optimization

Deterministic optimization



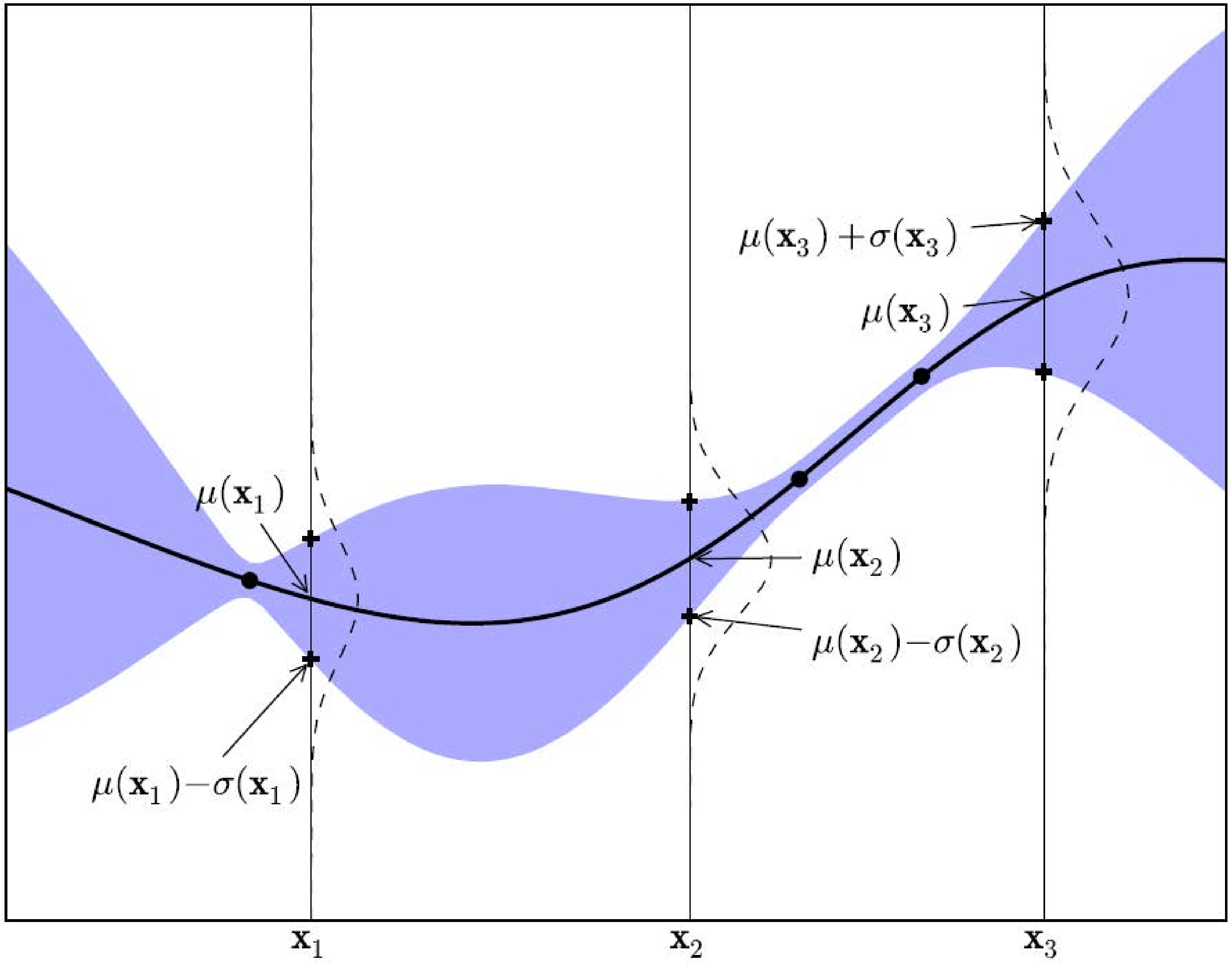


[Munos. From bandits to Monte-Carlo Tree Search: The optimistic principle applied to optimization and planning. Foundations and Trends in Machine Learning ’14]

Bayesian optimization

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')).$$

A GP is a distribution over functions, completely specified by its mean function and covariance function



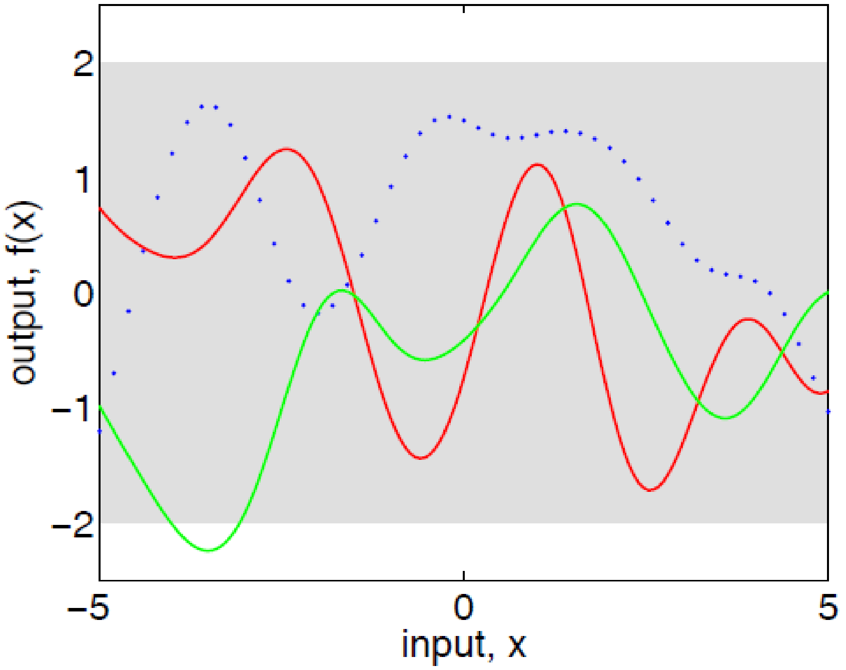
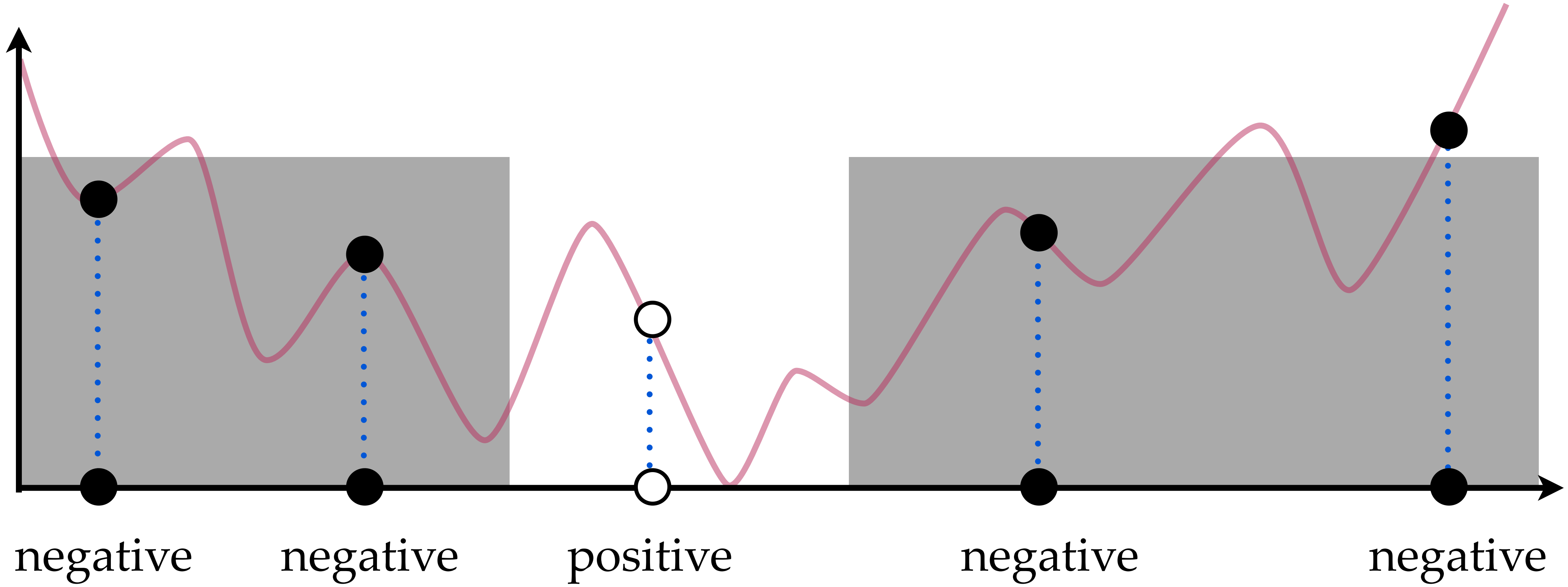


Figure 2: *Simple 1D Gaussian process with three observations. The solid black line is the GP surrogate mean prediction of the objective function given the data, and the shaded area shows the mean plus and minus the variance. The superimposed Gaussians correspond to the GP mean and standard deviation ($\mu(\cdot)$ and $\sigma(\cdot)$) of prediction at the points, $\mathbf{x}_{1:3}$.*

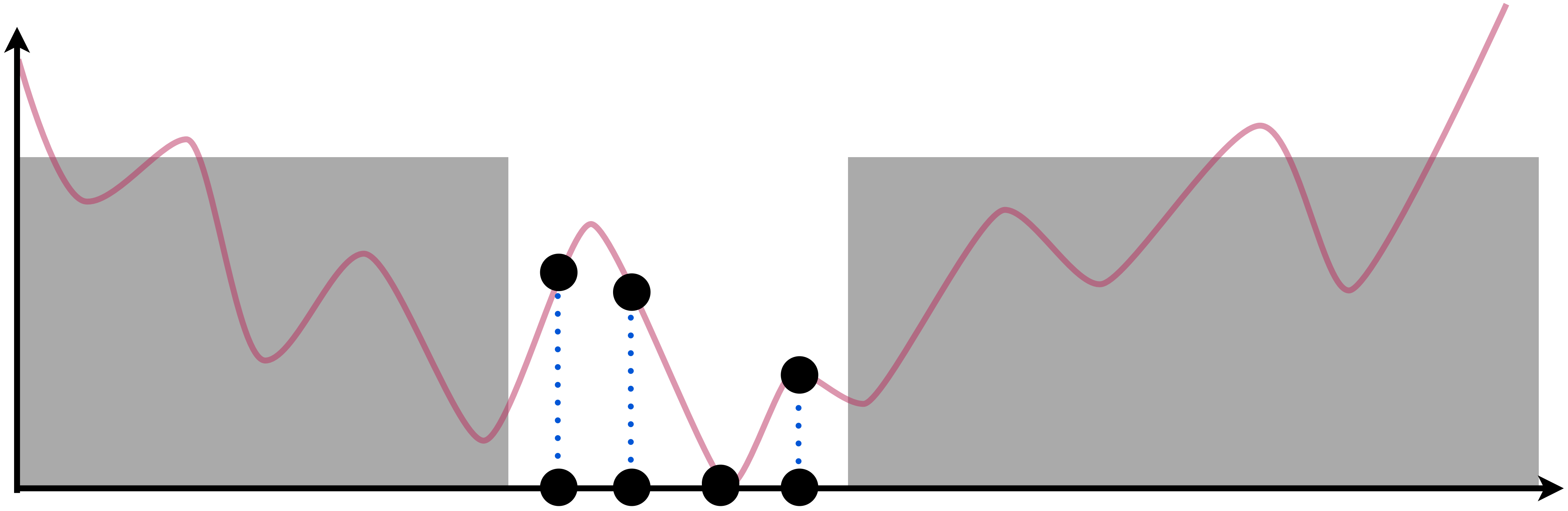
[Munos. From bandits to Monte-Carlo Tree Search: The optimistic principle applied to optimization and planning. Foundations and Trends in Machine Learning '14]

Classification-based optimization



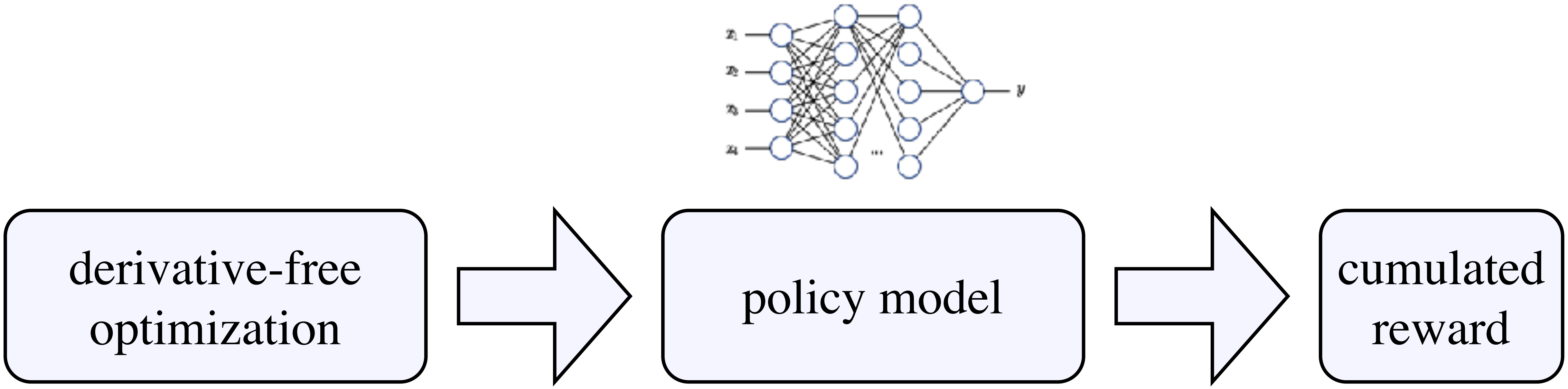
[Yu et al. Derivative-free optimization via classification. AAAI’16]

Classification-based optimization



[Yu et al. Derivative-free optimization via classification. AAAI’16]

Direct policy search



derivative-free optimization

policy model

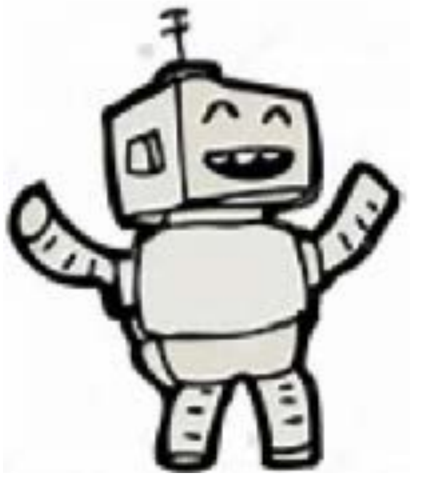
cumulated reward

converges slowly

usually good policy for complex tasks

Deep Reinforcement Learning

function approximation by
deep neural networks

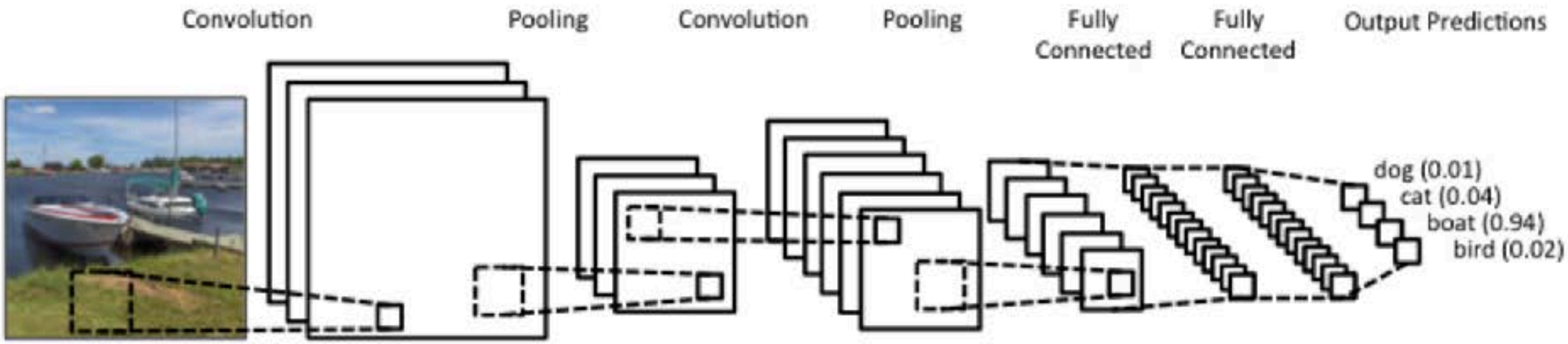


Convolutional neural networks

a powerful neural network architecture for image analysis

differentiable

require a lot of samples to train



Deep Q-Network

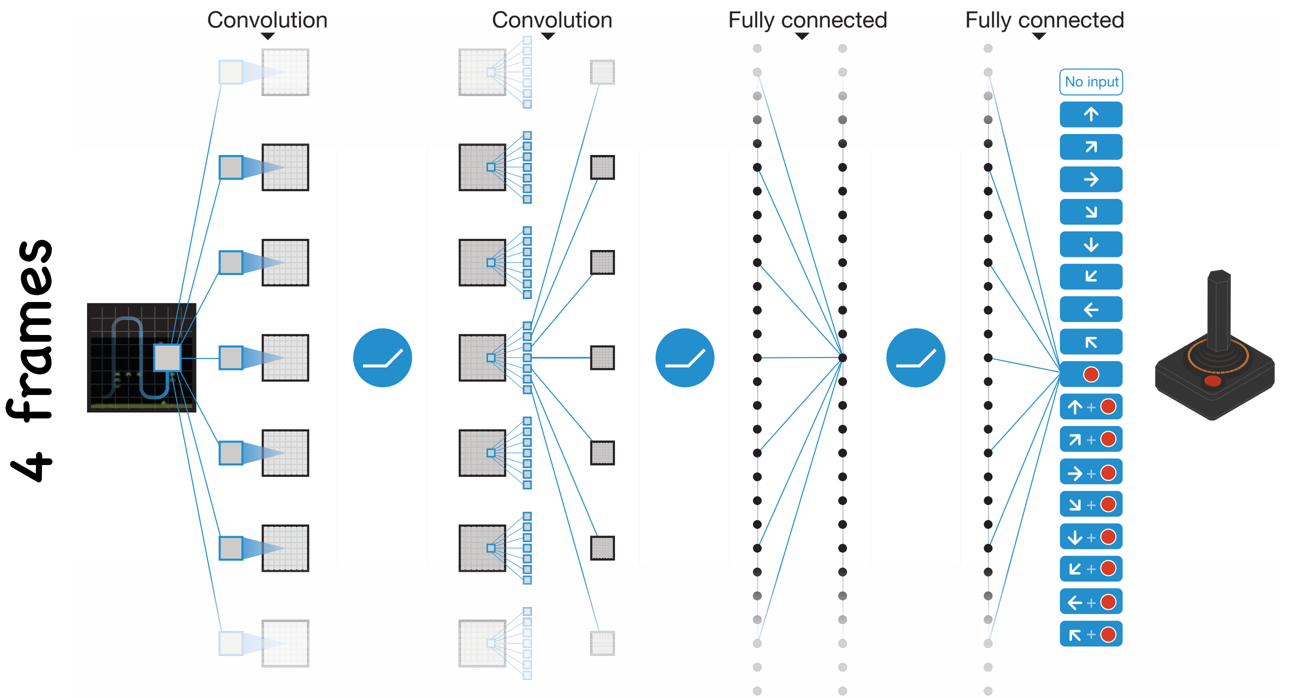
DQN

- using ϵ-greedy policy
- store 1million recent history (s,a,r,s’) in replay memory D
- sample a mini-batch (32) from D
- calculate Q-learning target $\tilde{Q}$
- update CNN by minimizing the Bellman error (delayed update)

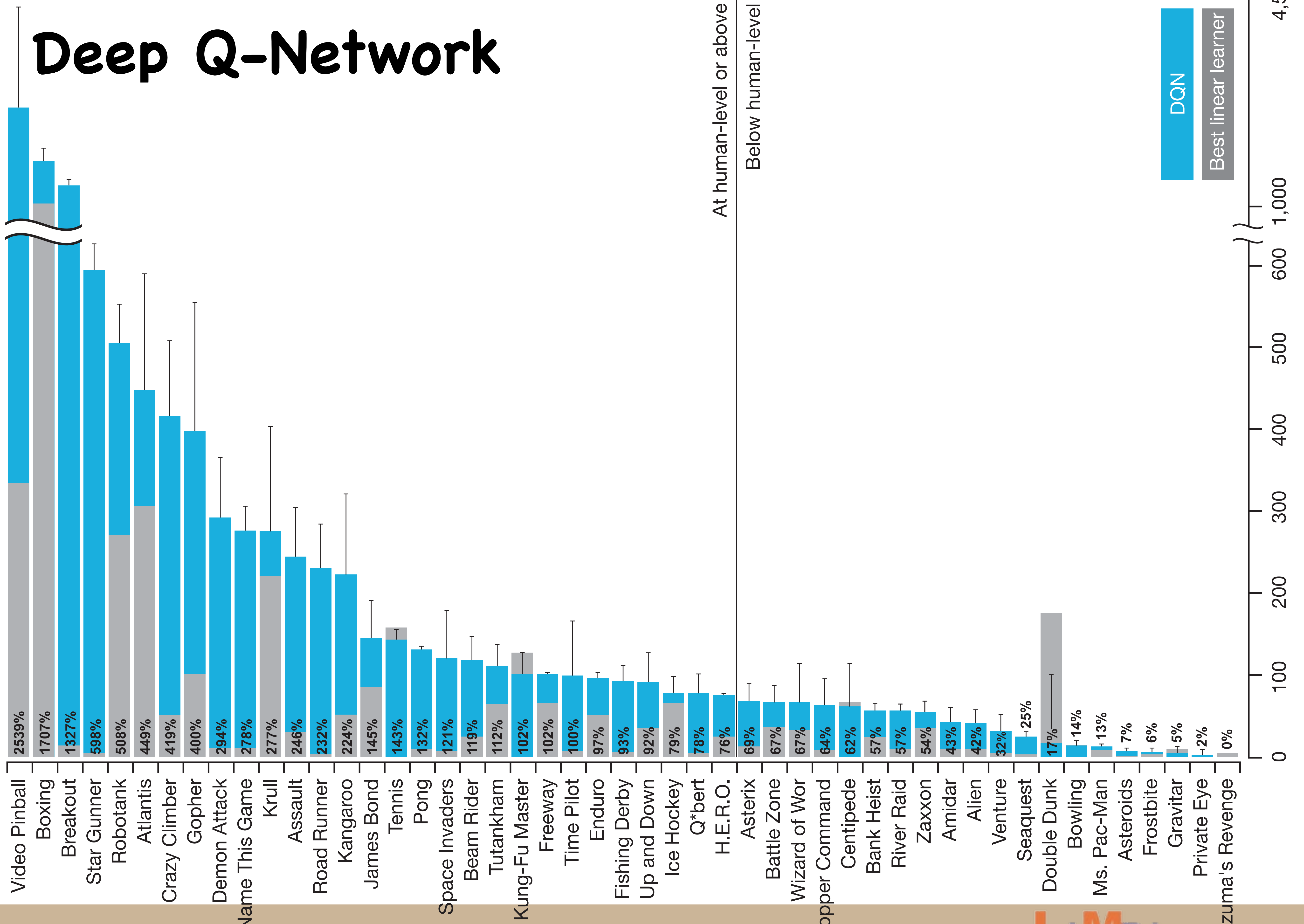
$$\sum (r + \gamma \max_{a'} \tilde{Q}(s', a') - Q_w(s, a))^2$$

DQN on Atari

learn to play from pixels



Deep Q-Network



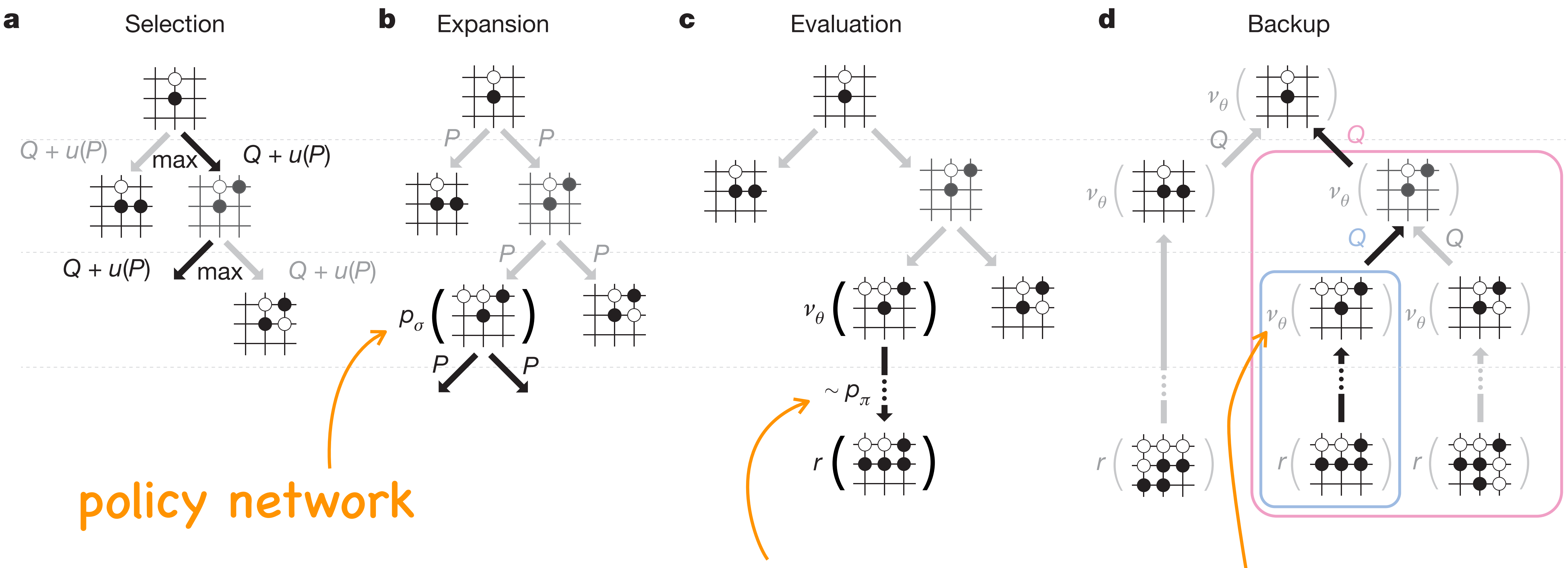
Deep Q-Network

effectiveness

Game	With replay, with target Q	With replay, without target Q	Without replay, with target Q	Without replay, without target Q
Breakout	316.8	240.7	10.2	3.2
Enduro	1006.3	831.4	141.9	29.1
River Raid	7446.6	4102.8	2867.7	1453.0
Seaquest	2894.4	822.6	1003.0	275.8
Space Invaders	1088.9	826.3	373.2	302.0

AlphaGo

A combination of tree search, deep neural networks and reinforcement learning



policy network

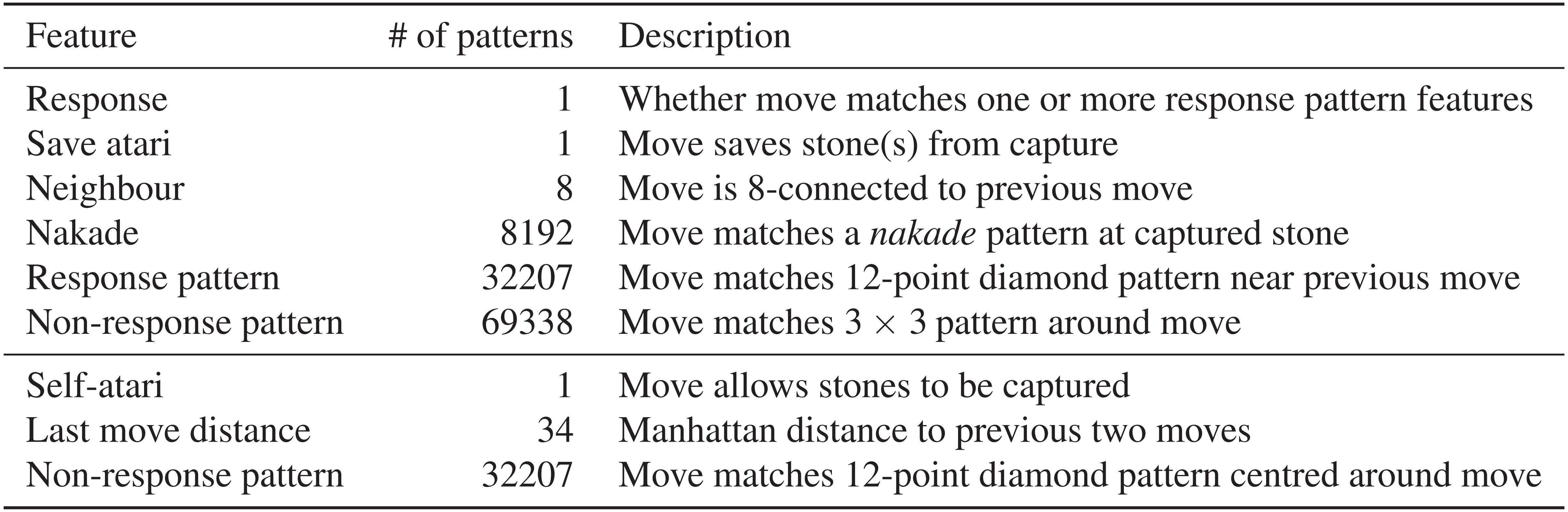
fast roll-out policy

value network

AlphaGo

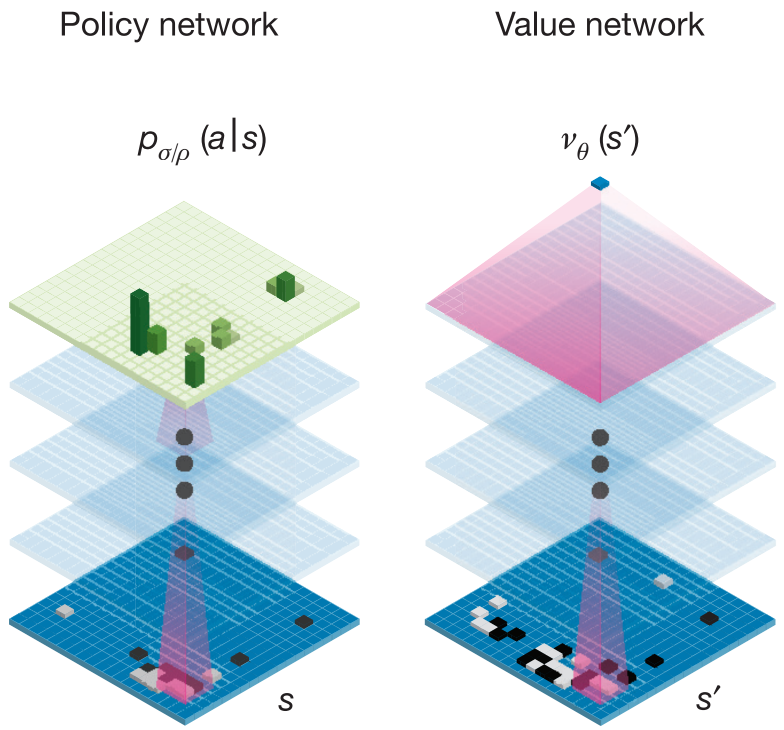
fast roll-out policy:

supervised learning from human v.s. human data



Feature	# of patterns	Description
Response	1	Whether move matches one or more response pattern features
Save atari	1	Move saves stone(s) from capture
Neighbour	8	Move is 8-connected to previous move
Nakade	8192	Move matches a *nakade* pattern at captured stone
Response pattern	32207	Move matches 12-point diamond pattern near previous move
Non-response pattern	69338	Move matches 3 × 3 pattern around move
Self-atari	1	Move allows stones to be captured
Last move distance	34	Manhattan distance to previous two moves
Non-response pattern	32207	Move matches 12-point diamond pattern centred around move

AlphaGo



policy network: a CNN output π(s,a)

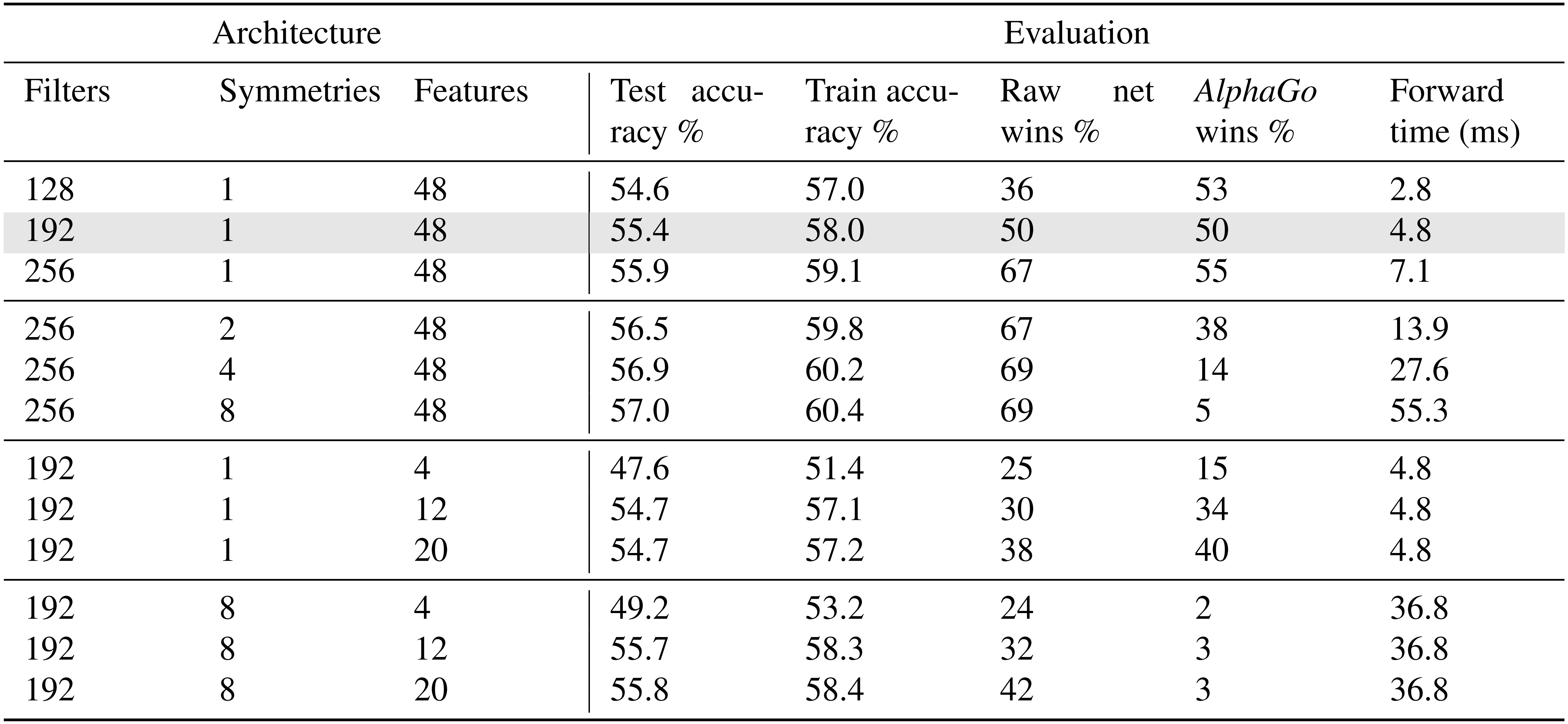
value network: a CNN output V(s)

Feature	# of planes	Description
Stone colour	3	Player stone / opponent stone / empty
Ones	1	A constant plane filled with 1
Turns since	8	How many turns since a move was played
Liberties	8	Number of liberties (empty adjacent points)
Capture size	8	How many opponent stones would be captured
Self-atari size	8	How many of own stones would be captured
Liberties after move	8	Number of liberties after this move is played
Ladder capture	1	Whether a move at this point is a successful ladder capture
Ladder escape	1	Whether a move at this point is a successful ladder escape
Sensibleness	1	Whether a move is legal and does not fill its own eyes
Zeros	1	A constant plane filled with 0
Player color	1	Whether current player is black

AlphaGo

policy network: initialization

supervised learning from human v.s. human data

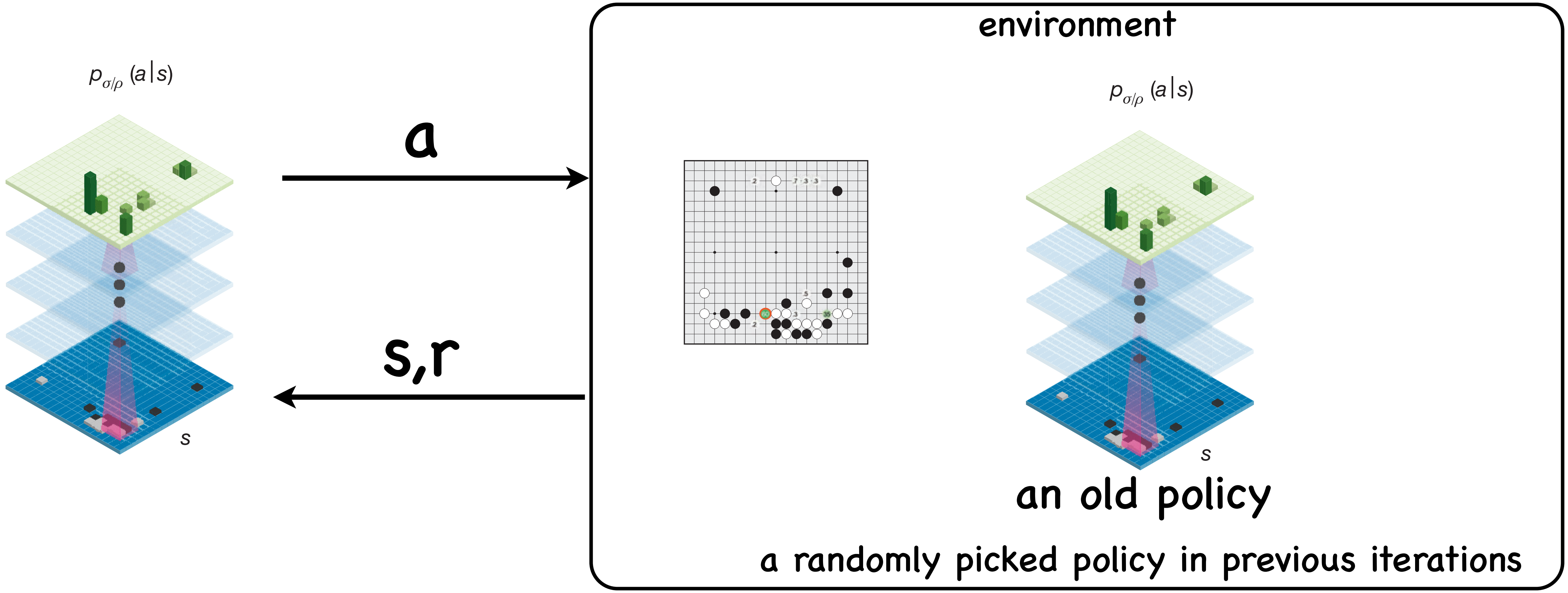


Architecture			Evaluation				
Filters	Symmetries	Features	Test accuracy %	Train accuracy %	Raw net wins %	*AlphaGo* wins %	Forward time (ms)
128	1	48	54.6	57.0	36	53	2.8
192	1	48	55.4	58.0	50	50	4.8
256	1	48	55.9	59.1	67	55	7.1
256	2	48	56.5	59.8	67	38	13.9
256	4	48	56.9	60.2	69	14	27.6
256	8	48	57.0	60.4	69	5	55.3
192	1	4	47.6	51.4	25	15	4.8
192	1	12	54.7	57.1	30	34	4.8
192	1	20	54.7	57.2	38	40	4.8
192	8	4	49.2	53.2	24	2	36.8
192	8	12	55.7	58.3	32	3	36.8
192	8	20	55.8	58.4	42	3	36.8

AlphaGo

policy network: further improvement

reinforcement learning



a

s,r

environment

an old policy

a randomly picked policy in previous iterations

a.k.a. self-play

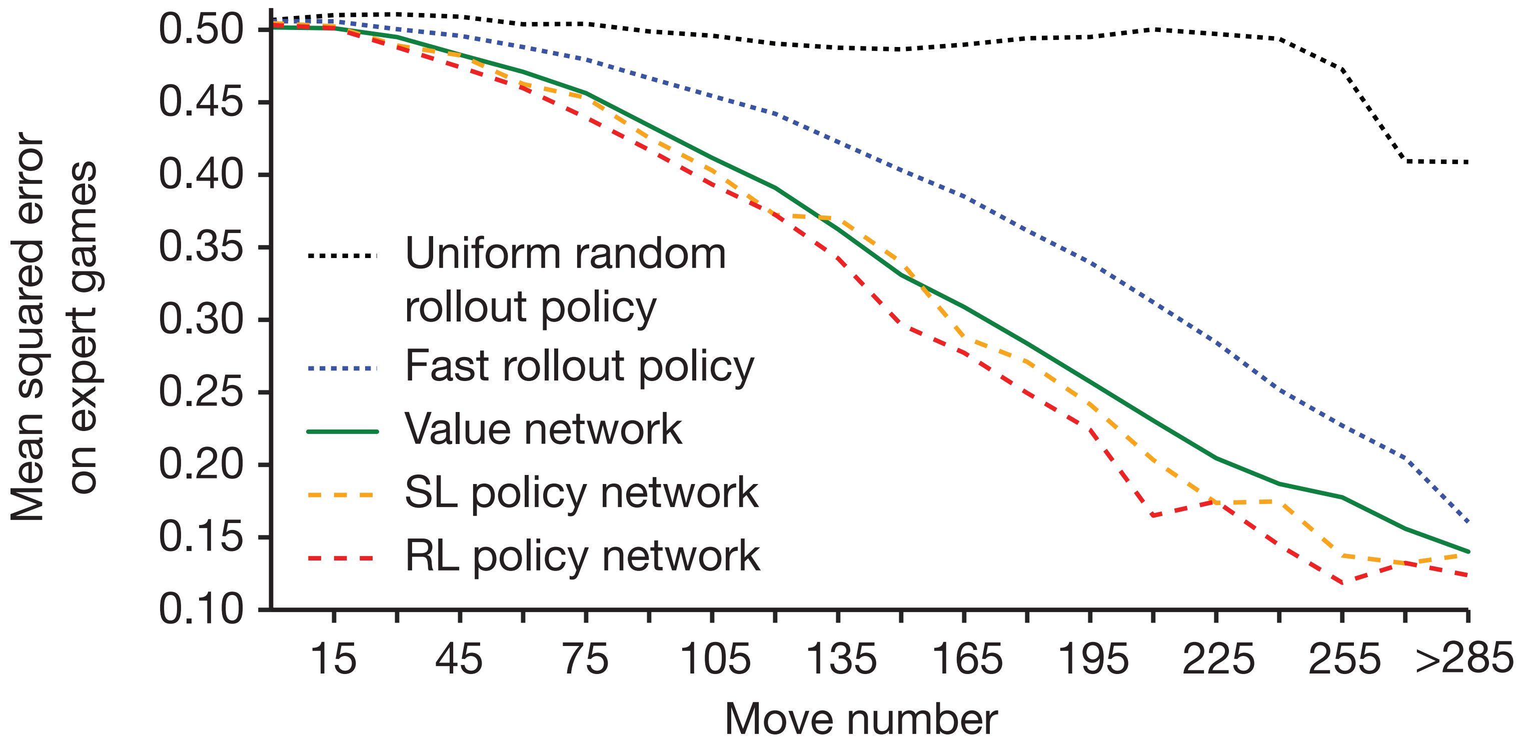
reward:

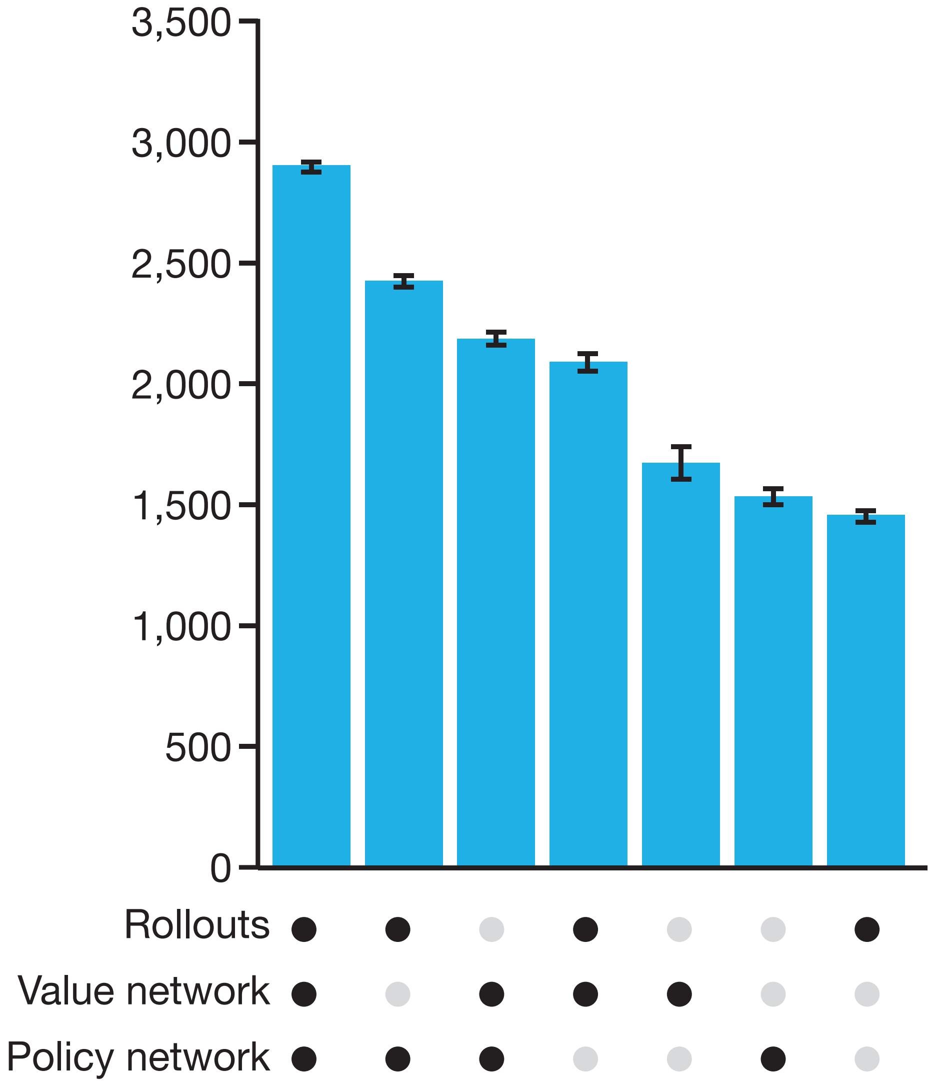
+1 -- win at terminate state

-1 -- loss at terminate state

AlphaGo

value network: supervised learning from RL data





Other directions

- Partial-observable and other semi-MDP
- Learning from demonstrations
- Transfer learning in reinforcement learning
- …

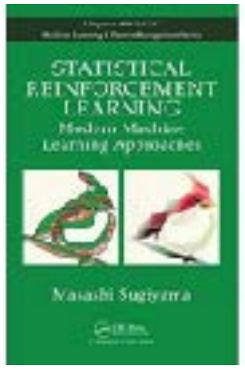
Robot Motor Skill Coordination with EM-based Reinforcement Learning

Petar Kormushev, Sylvain Calinon, and Darwin G. Caldwell

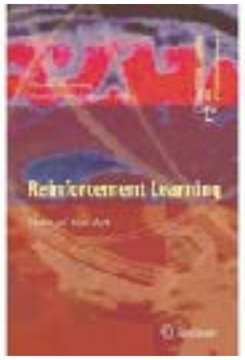
Italian Institute of Technology

Books

Richard S. Sutton and Andrew G. Barto
Reinforcement Learning: An Introduction


STATISTICAL REINFORCEMENT LEARNING
Modern Machine Learning Approaches
Masashi Sugiyama

Masashi Sugiyama
Statistical Reinforcement Learning:
Modern Machine Learning Approaches



Marco Wiering and Martijn van Otterlo (eds)
Reinforcement Learning: State-of-the-Art

Also in MDP books

Mykel J. Kochenderfer
Decision Making Under Uncertainty:
Theory and Application

and machine learning books

周志华
机器学习

Venues

AI journal, JAIR, JMLR, ML journal, …

IJCAI, AAAI, ICML, NIPS, AAMAS, IROS, …

IJCAI-17

MELBOURNE

19-25 AUGUST 2017

MELBOURNE CONVENTION CENTRE

IMPORTANT DATES

Abstract submission: February 16th, 2017 // **Paper submission:** February 19th, 2017