强化学习

与生成对抗网络(GAN)入门

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

俞扬

南京大学 计算机系
软件新技术国家重点实验室
机器学习与数据挖掘研究所 (LAMDA)

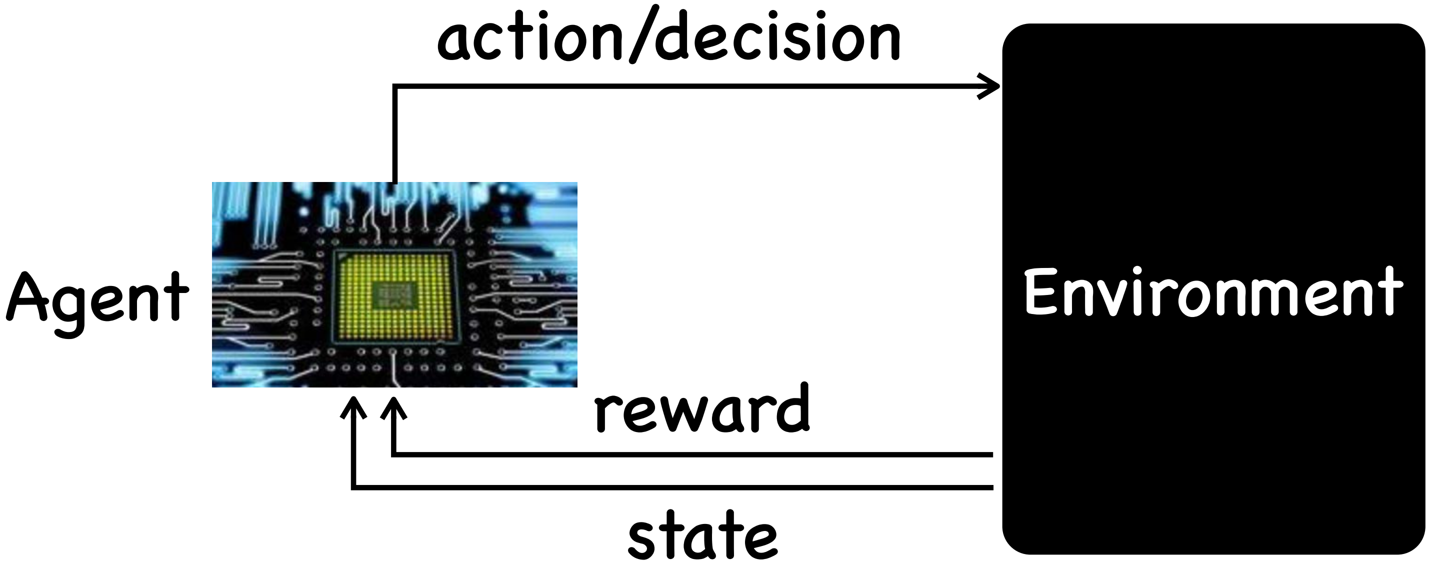
Reinforcement learning

intelligent animals can learn from interactions

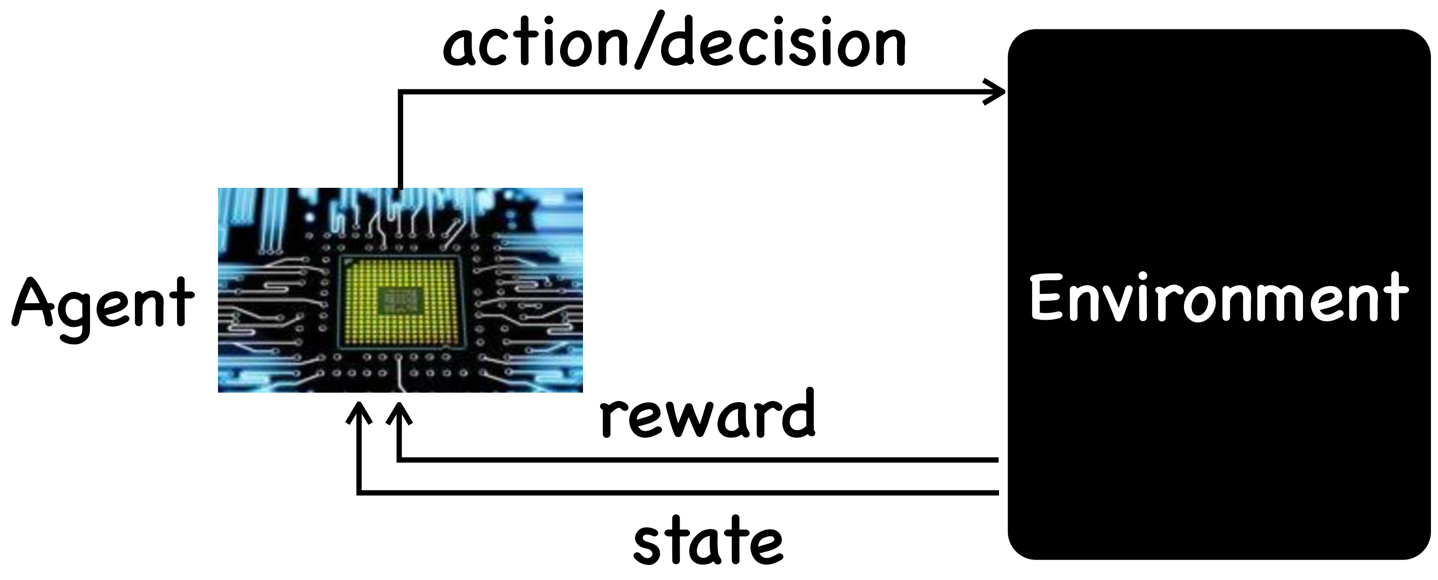
to adapt to the environment

can computers do similarly?

reinforcement learning, the "real artificial intelligence"



Reinforcement learning



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

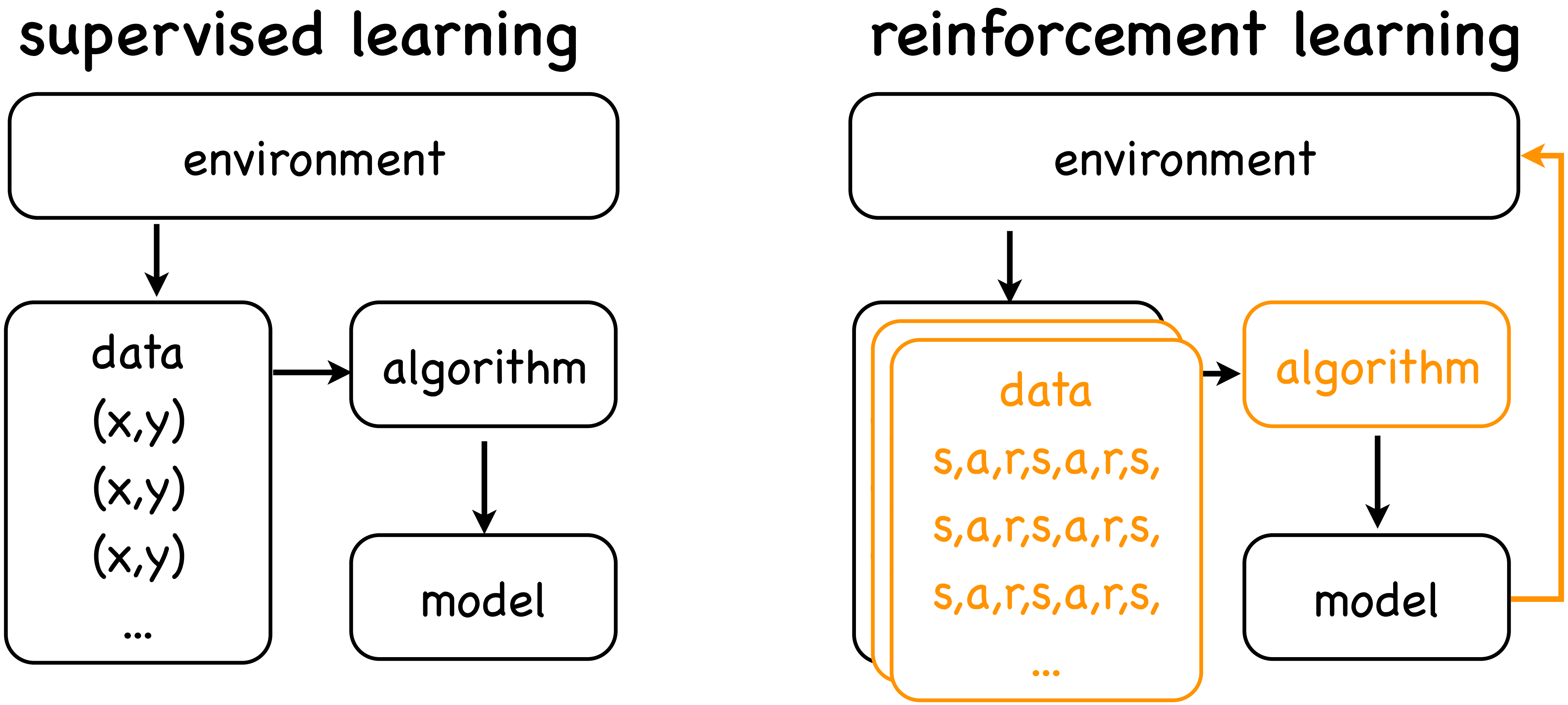
Policy (deterministic): $\pi : S \to A$

Agent's goal: learn a policy to maximize long-term total reward

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

Difference between RL and SL?

both learn a model …



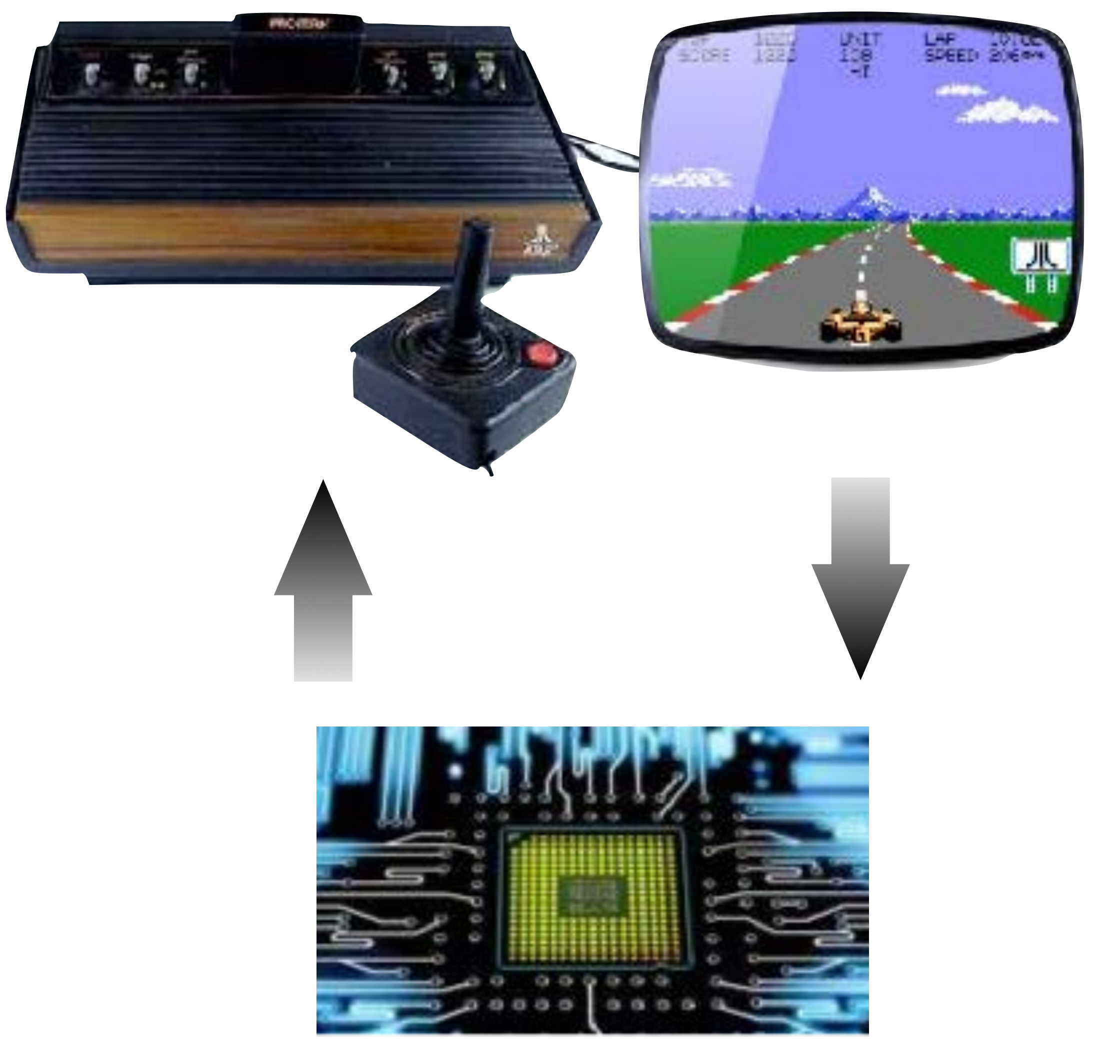
open loop
learning from labeled data
passive data

closed loop
learning from delayed reward
explore environment

Applications: The Atari games

Deepmind Deep Q-learning on Atari

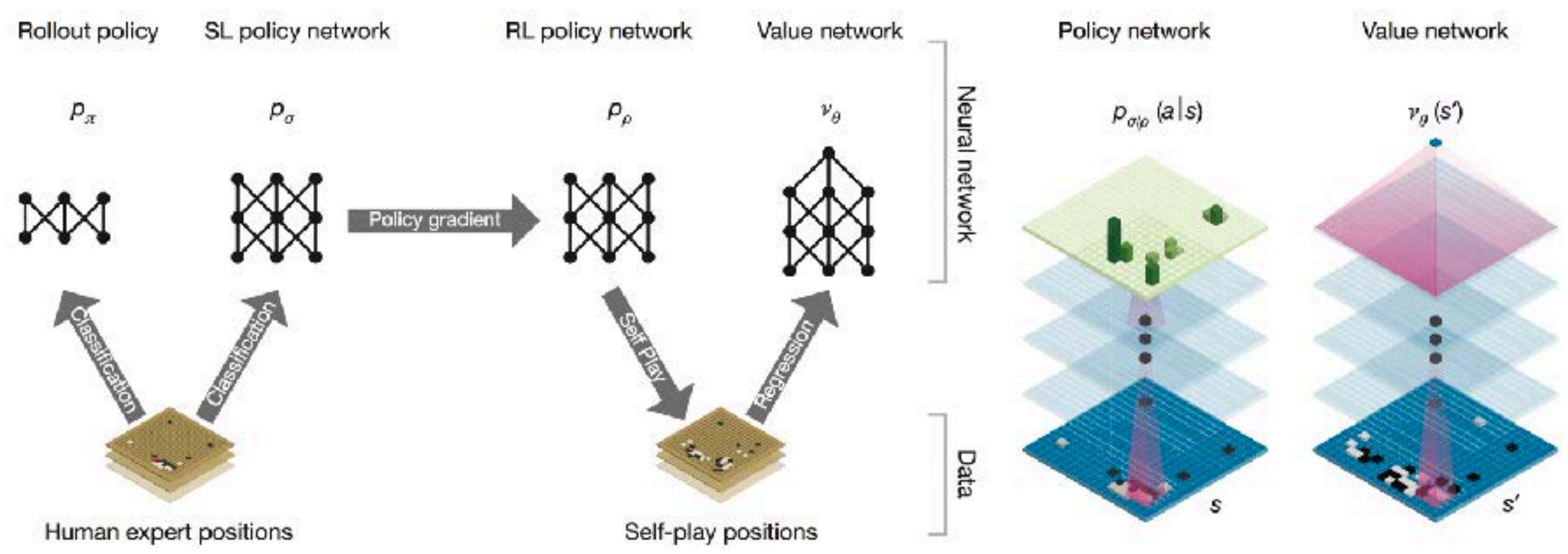
[Mnih *et al*. Human-level control through deep reinforcement learning. Nature, 518(7540): 529-533, 2015]



Applications: The game of Go

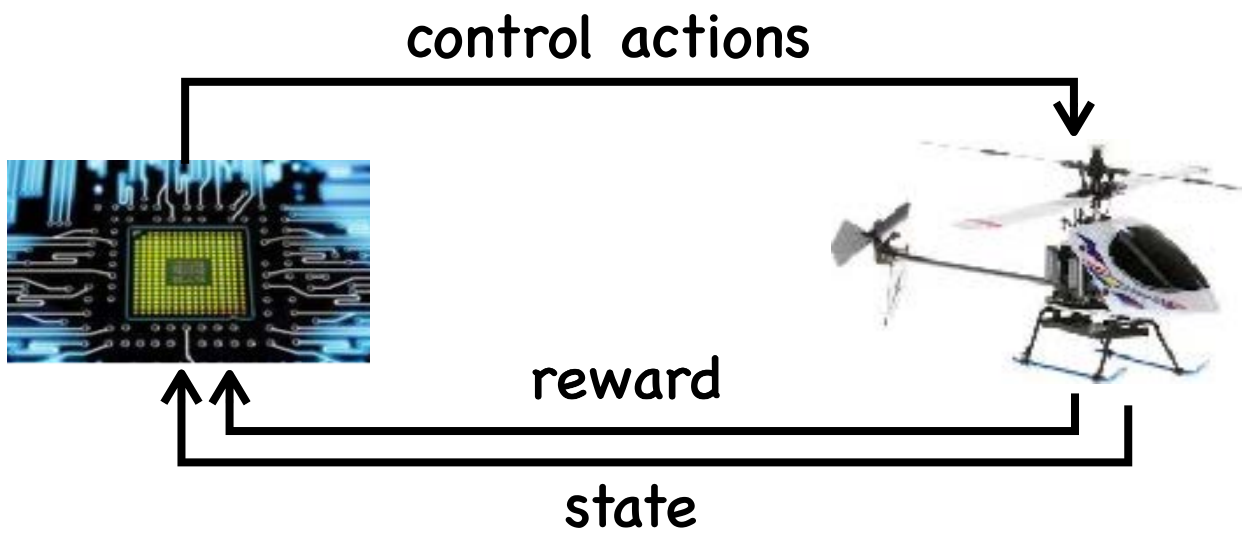
Deepmind AlphaGo system

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



Applications: Robotics

learning robot skills



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

More applications

Search

Recommendation system

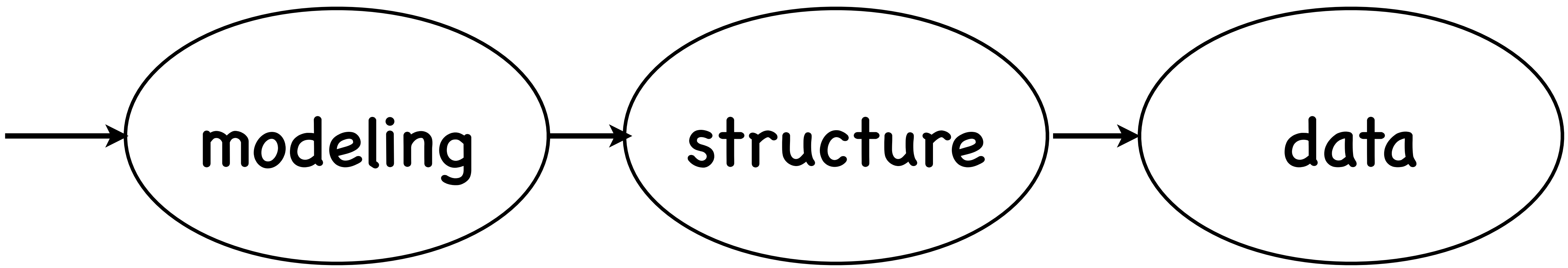
Stock prediction

...

every decision changes the world

More applications

Also as an differentiable approach for



[Bahdanau et al., An Actor-Critic Algorithm for Sequence Prediction. ArXiv 1607.07086]

[He et al., Deep Reinforcement Learning with a Natural Language Action Space, ACL'16]

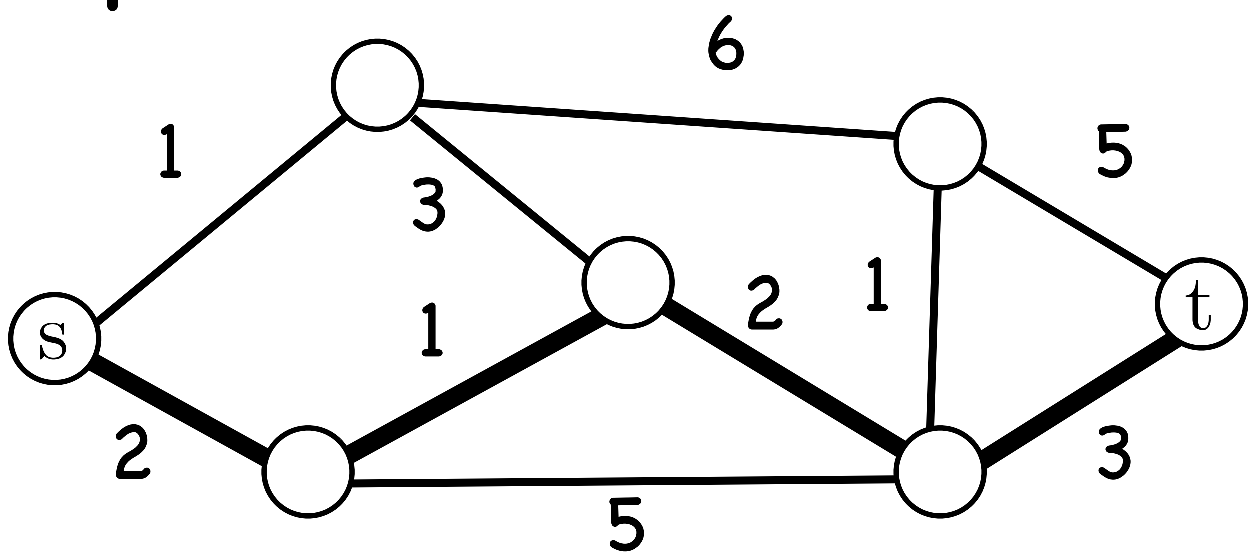
[B. Dhingra et al., End-to-End Reinforcement Learning of Dialogue Agents for Information Access, ArXiv 1609.00777]

[Yu et al., SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient, AAAI'17]

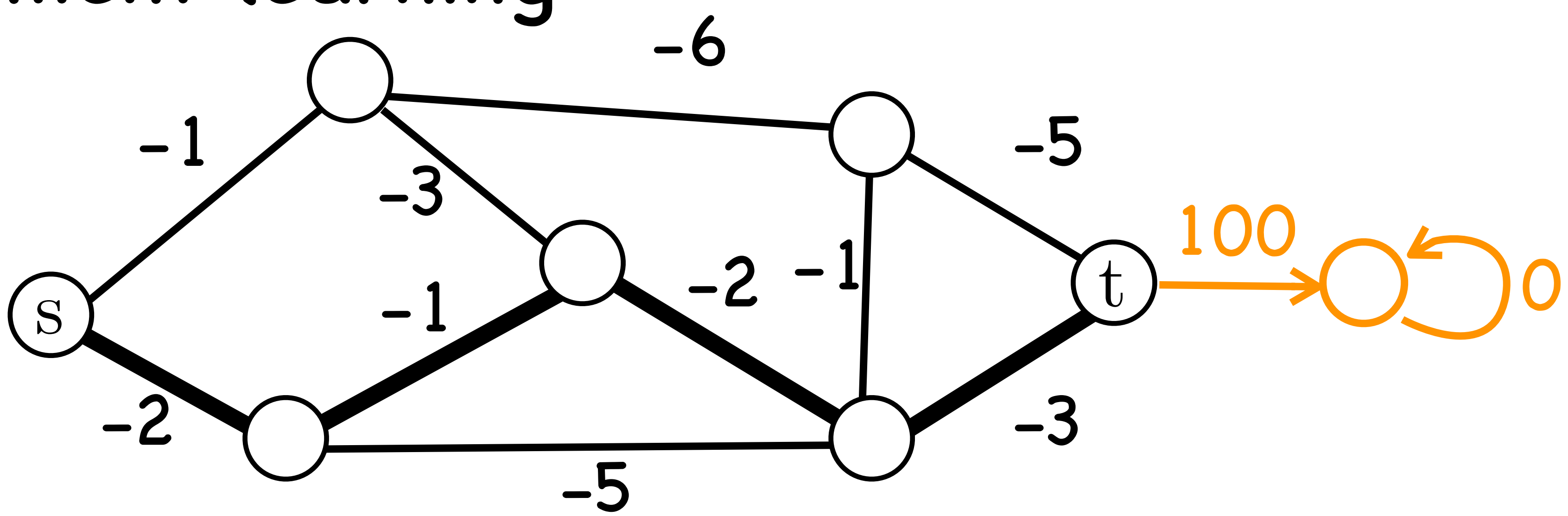
...

Generality of RL

shortest path problem:



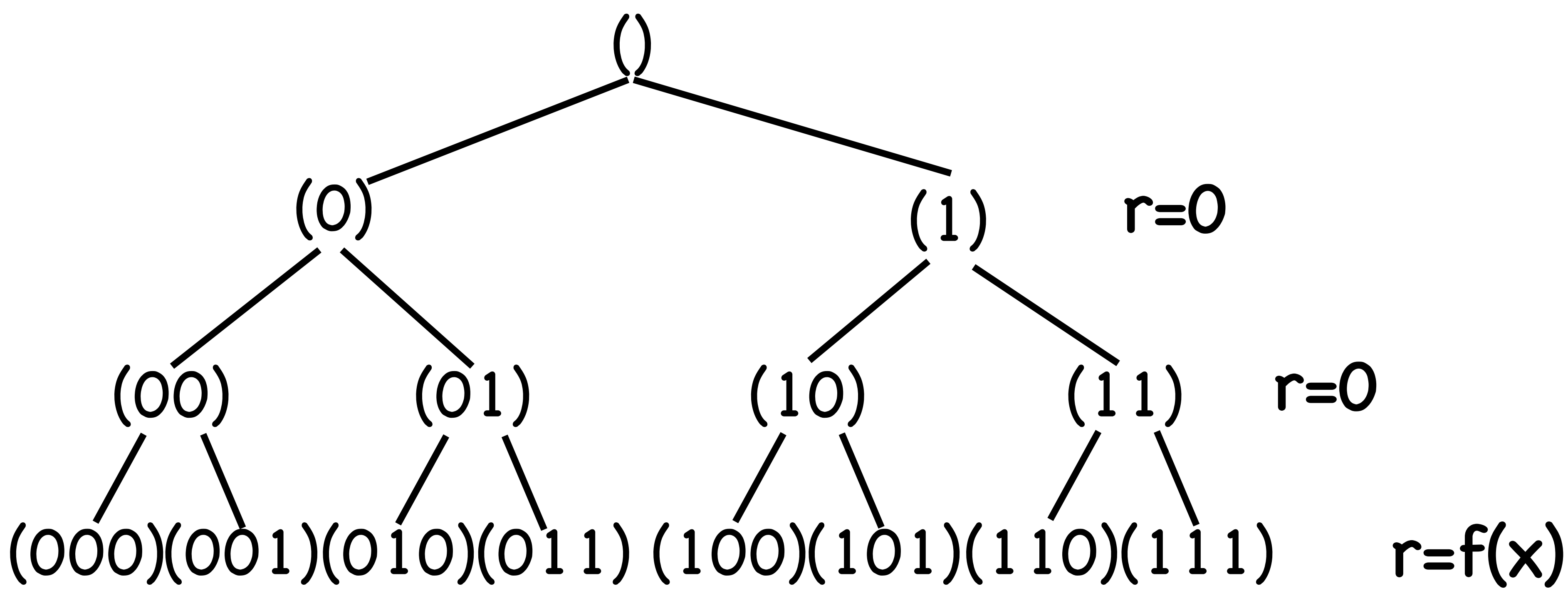
by reinforcement learning



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

Generality of RL

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



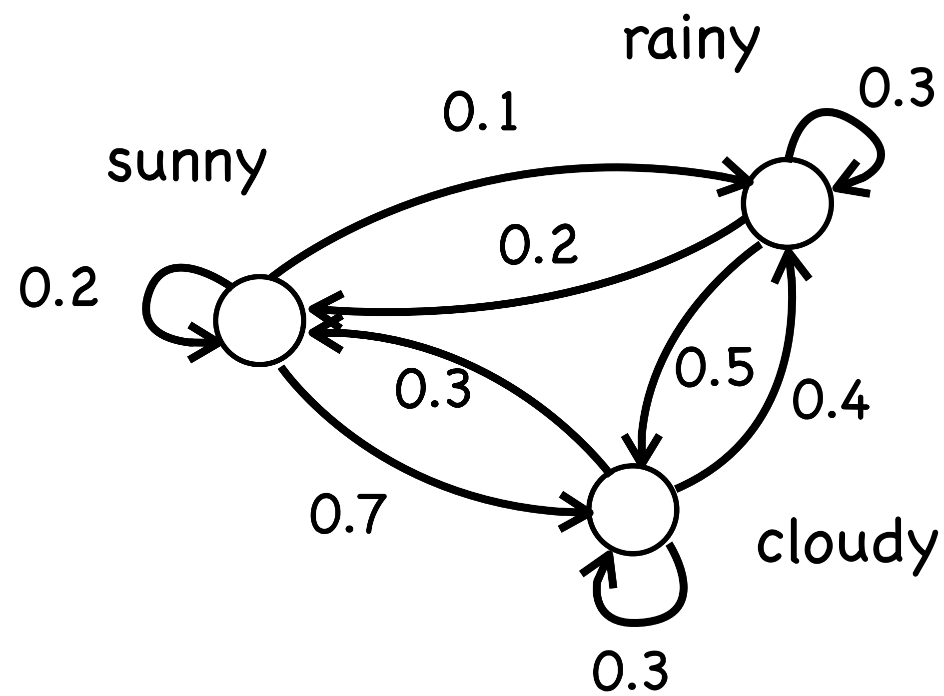
solving the optimal policy is NP-hard!

do not formulate everything as RL problem

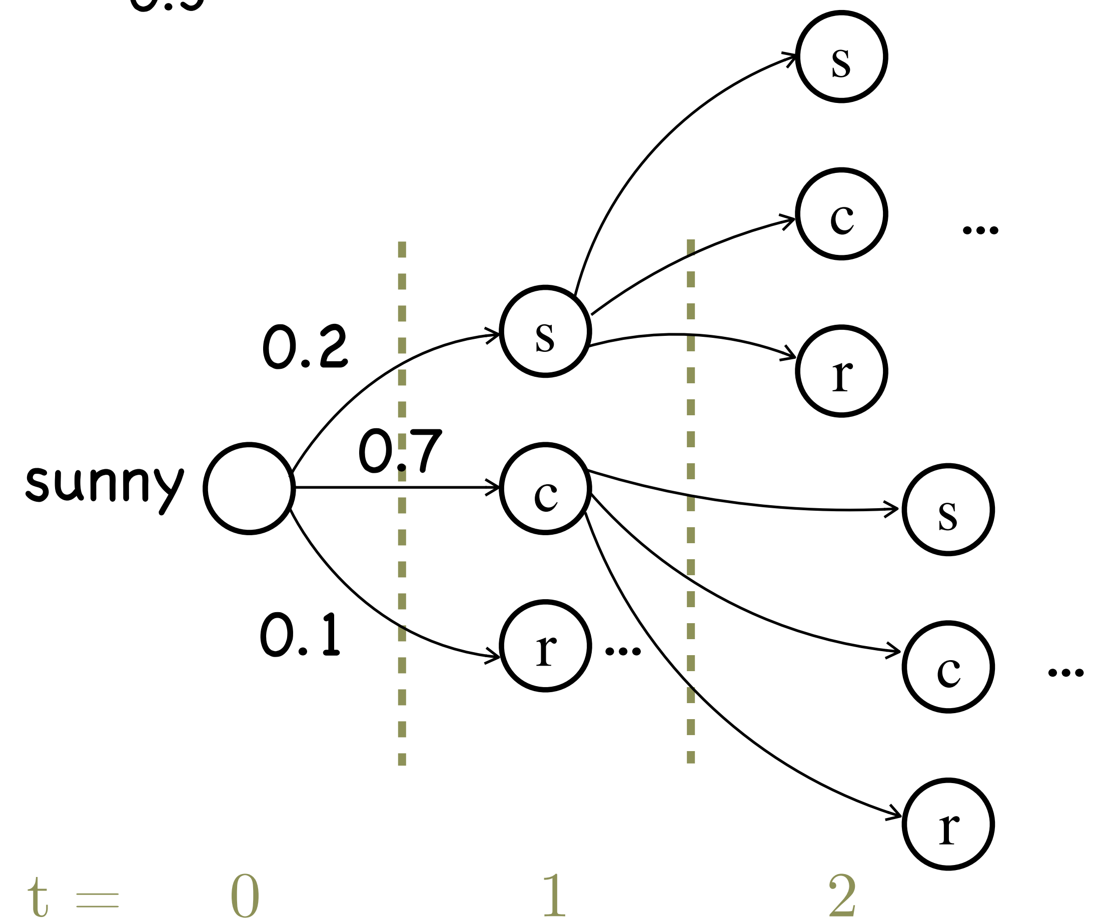
Outline

- Markov Decision Process
- Value-based methods
- Policy search
- Deep reinforcement learning
- Imitation learning (<- GAN is here)
- Discussion on the future

From Markov Process to MDP

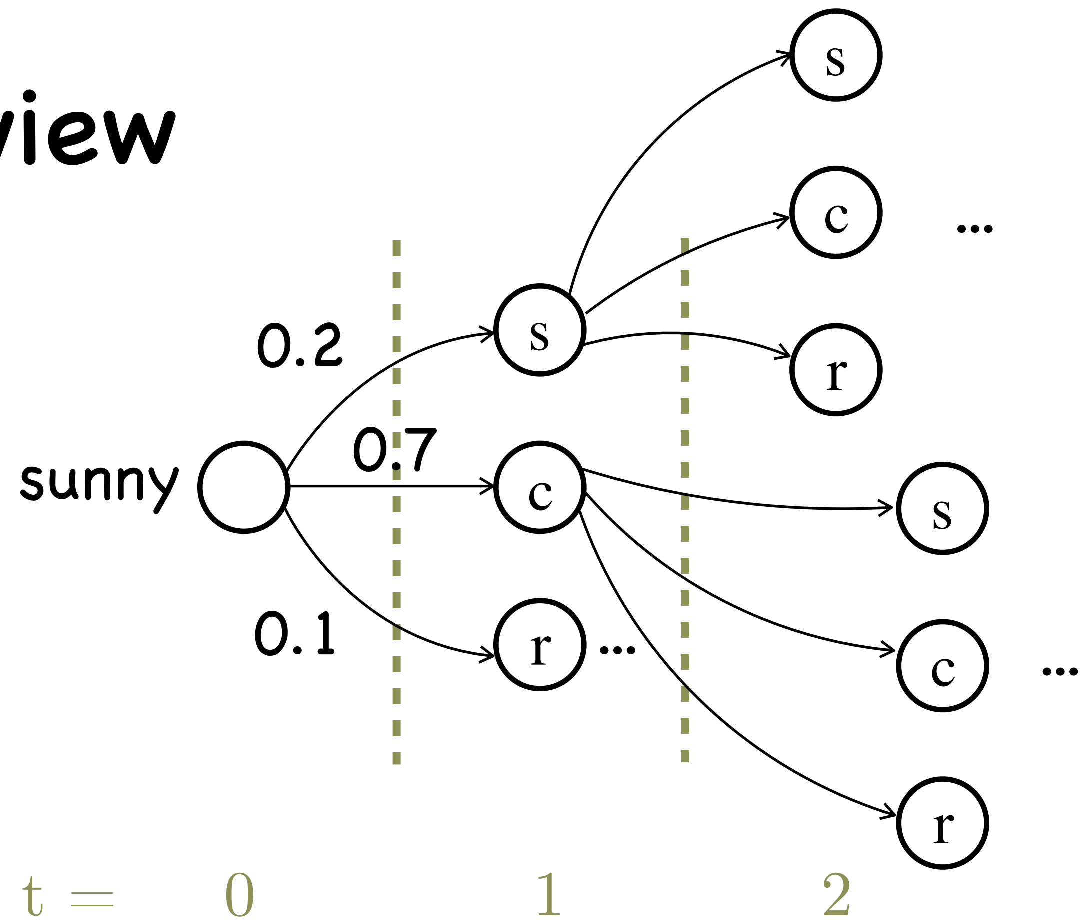


horizontal view

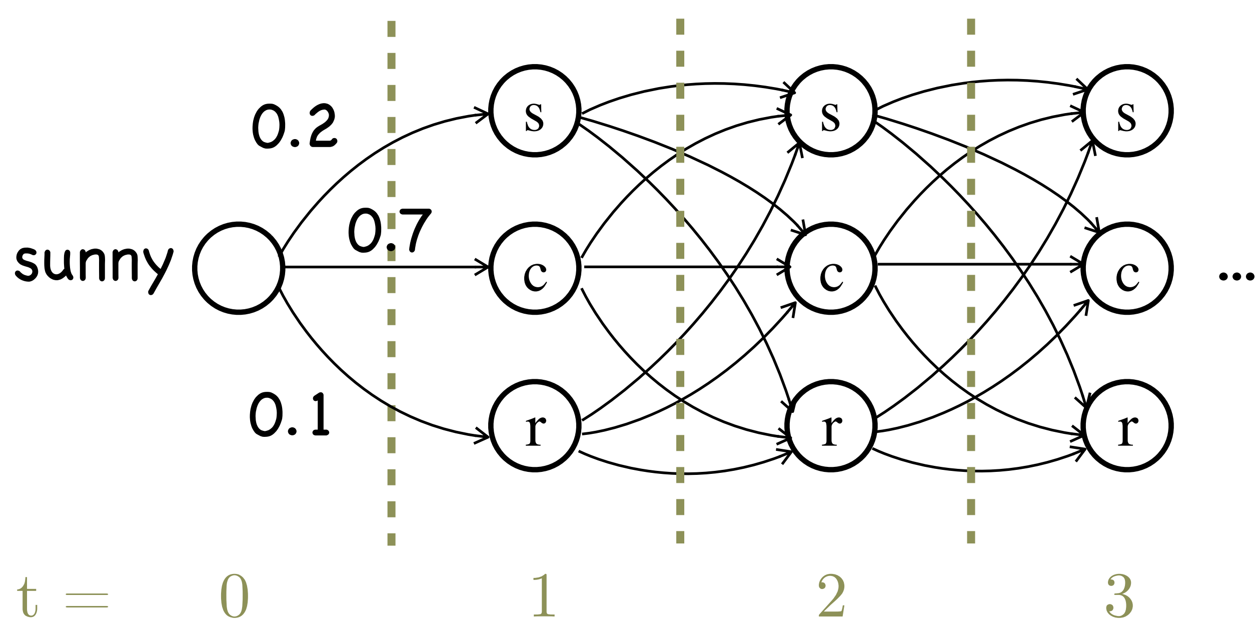


From Markov Process to MDP

horizontal view

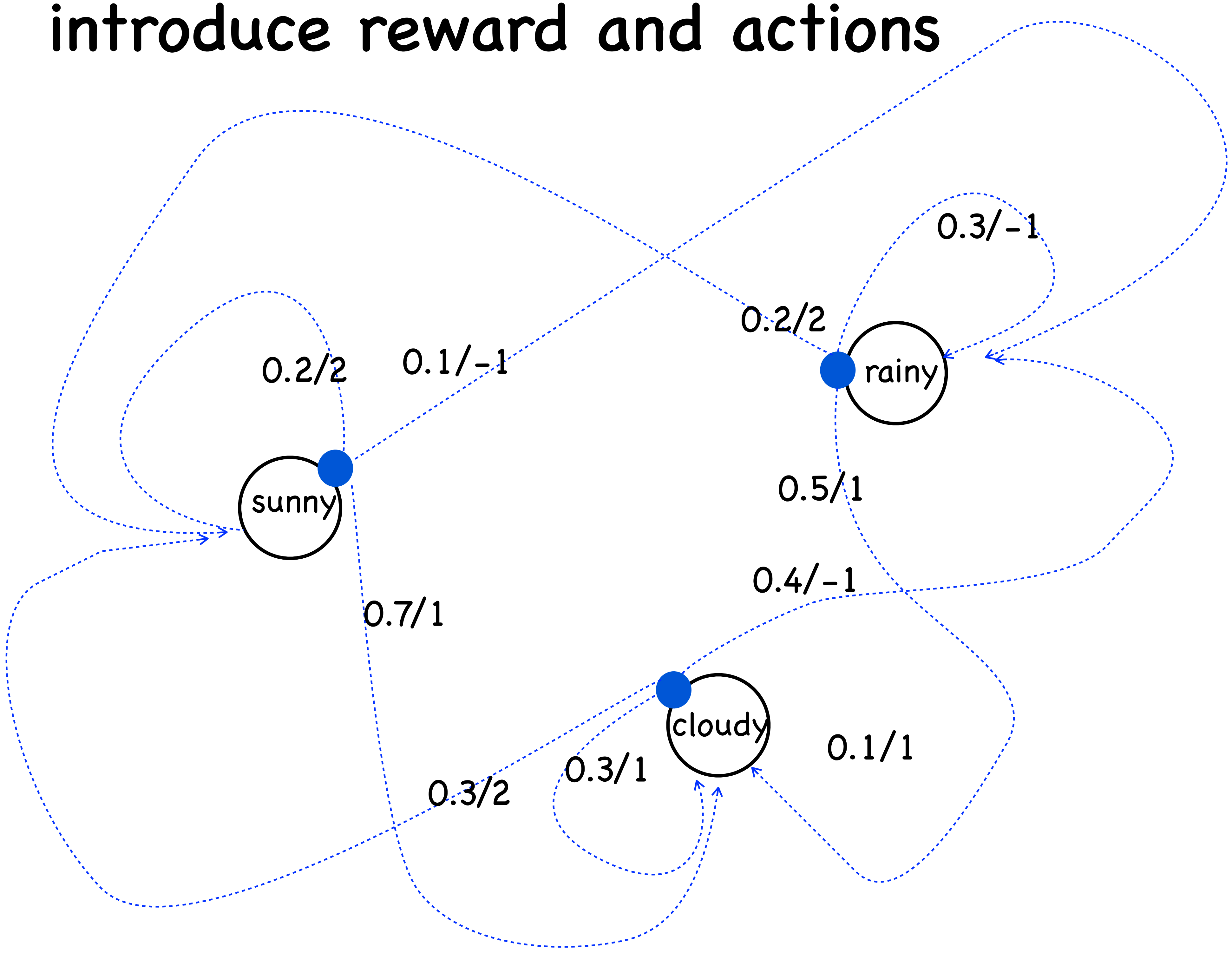


compactly



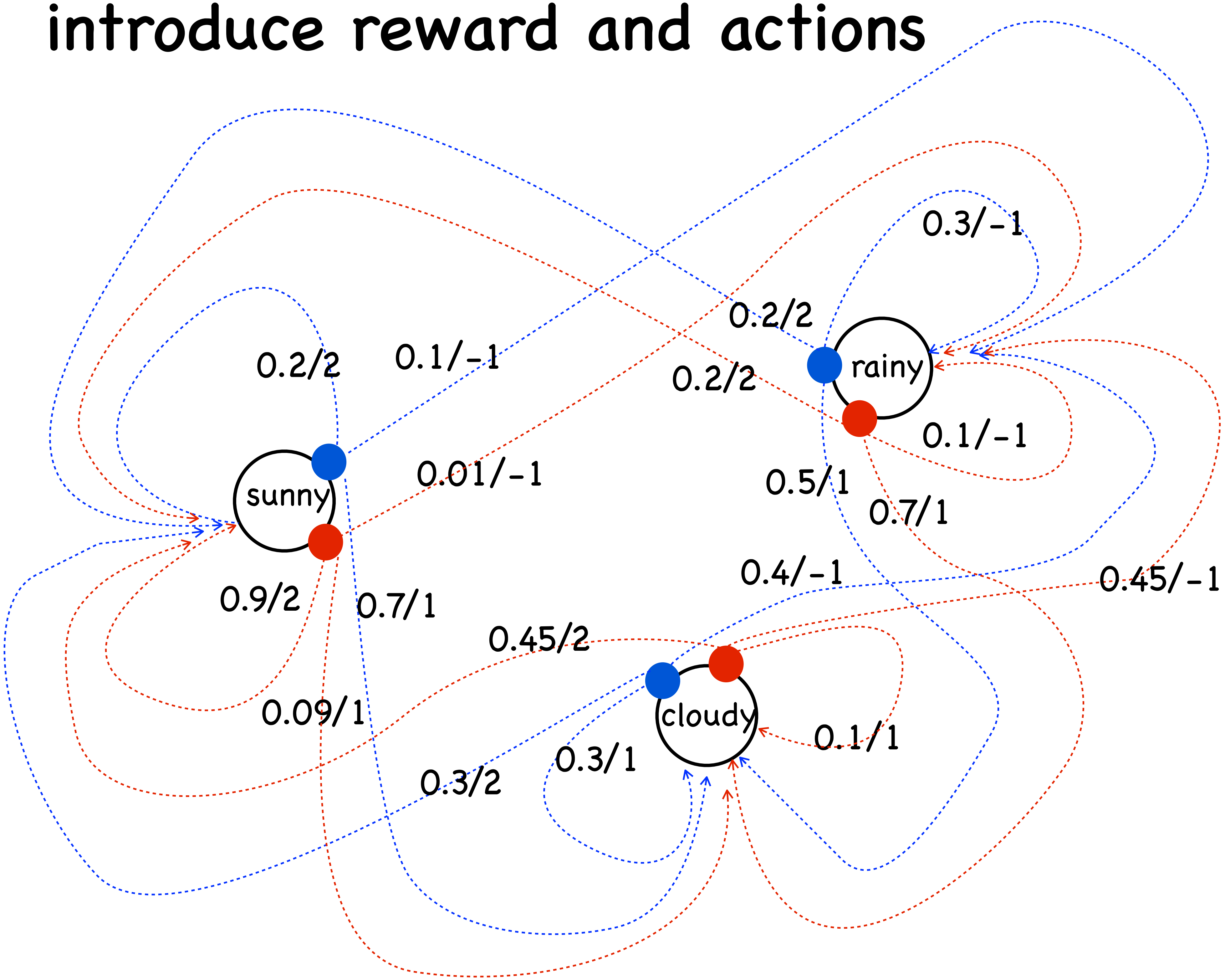
From Markov Process to MDP

introduce reward and actions



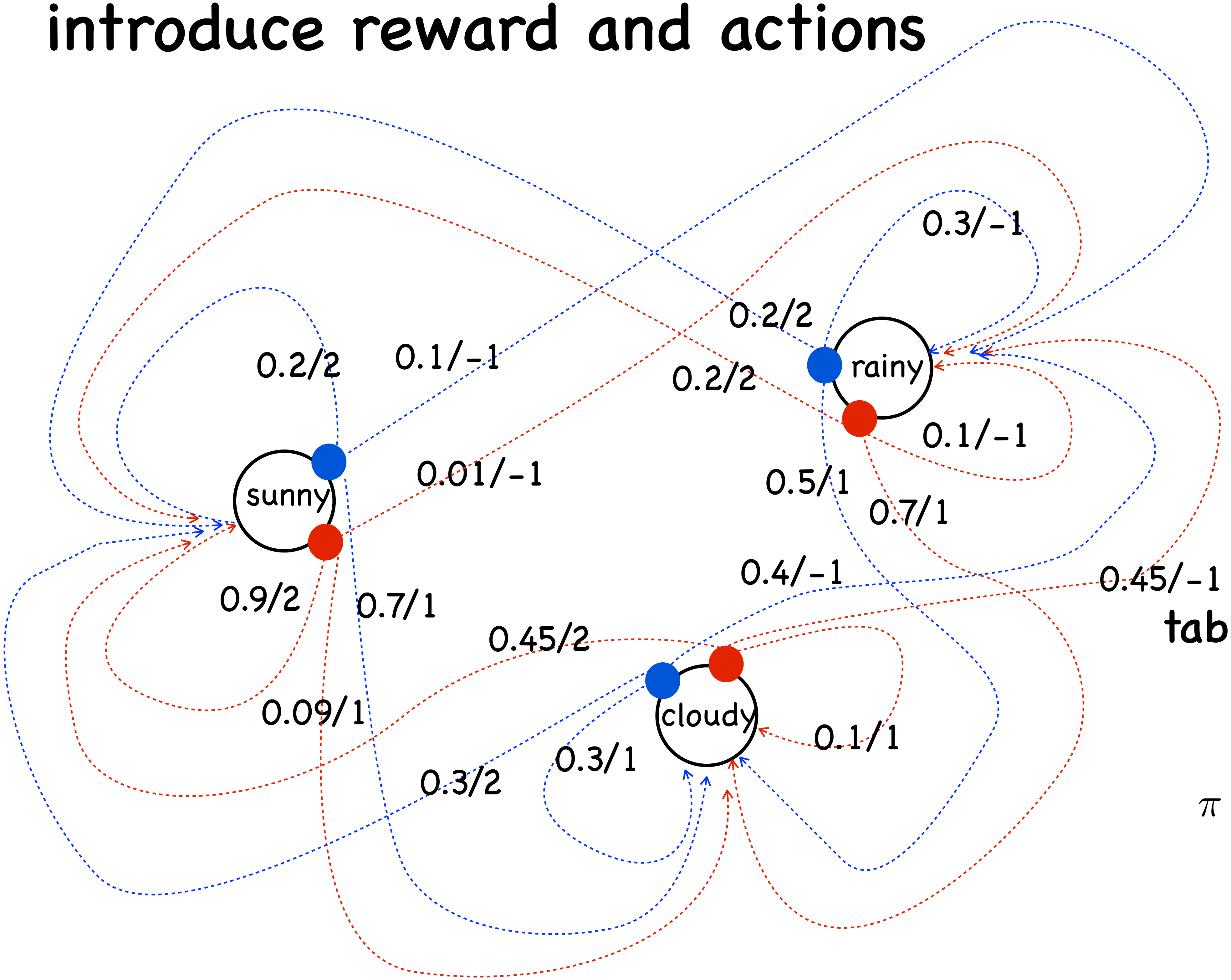
From Markov Process to MDP

introduce reward and actions



From Markov Process to MDP

introduce reward and actions



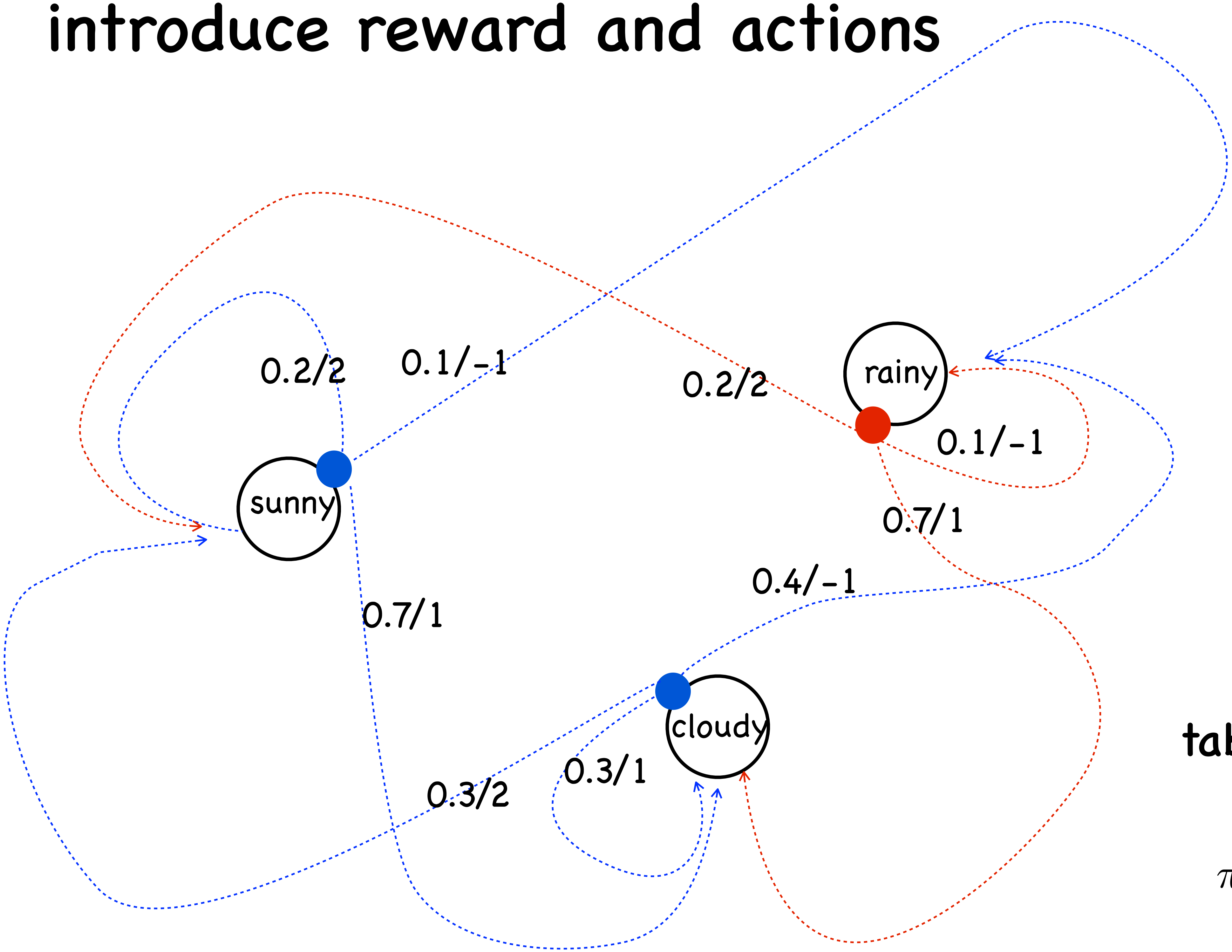
tabular representation

$\pi =$

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

From Markov Process to MDP

introduce reward and actions



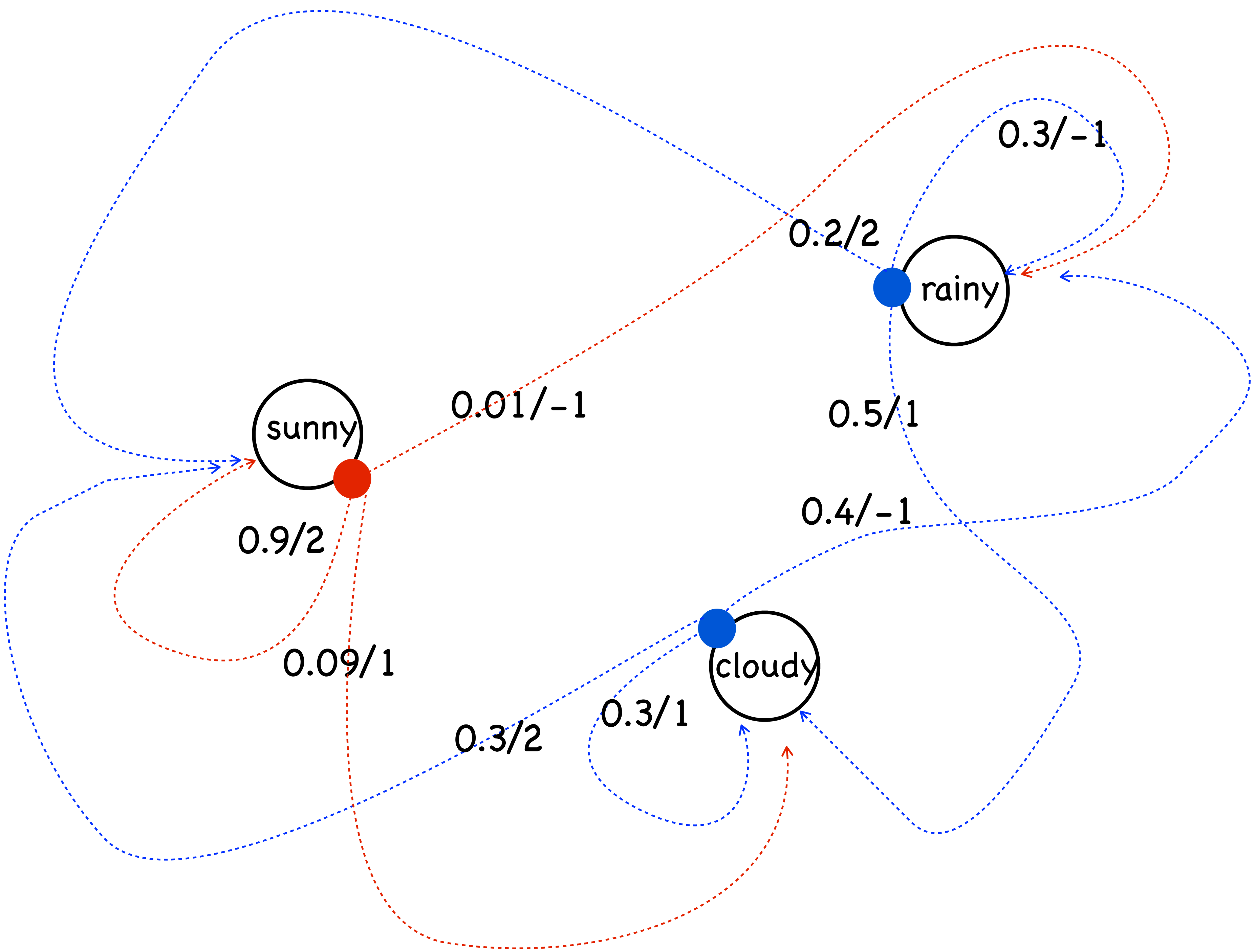
tabular representation

$\pi =$

s	0
c	0
r	1

From Markov Process to MDP

introduce reward and actions



tabular representation

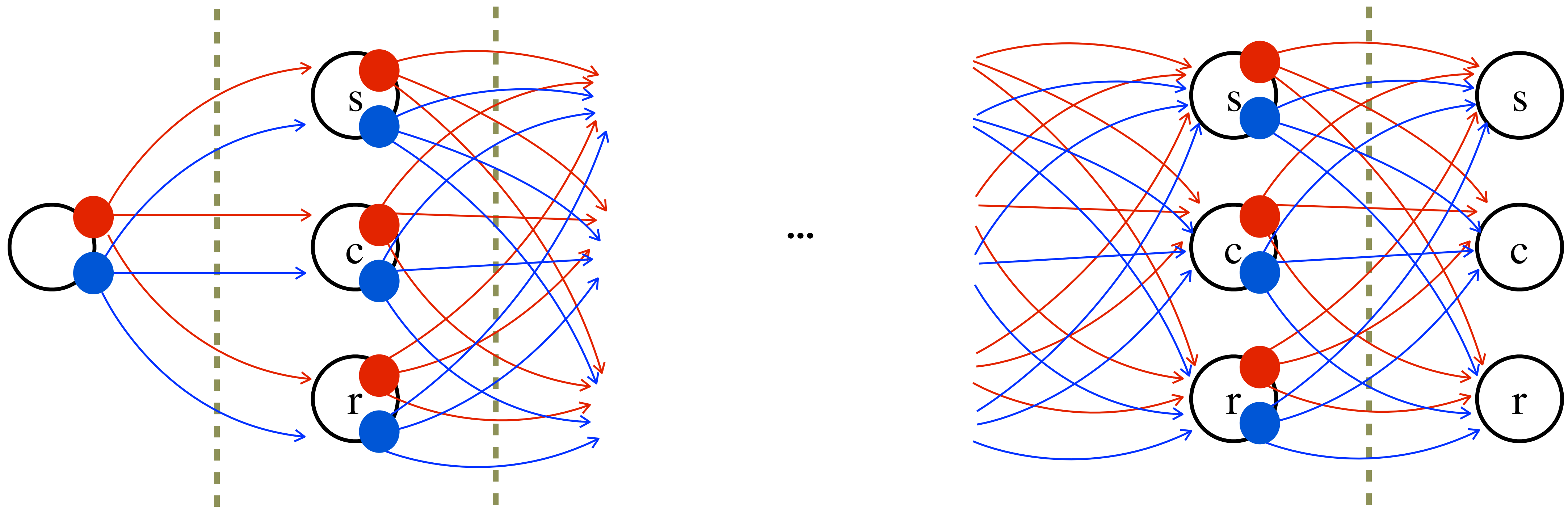
$\pi =$

s	1
c	0
r	0

Markov Decision Process

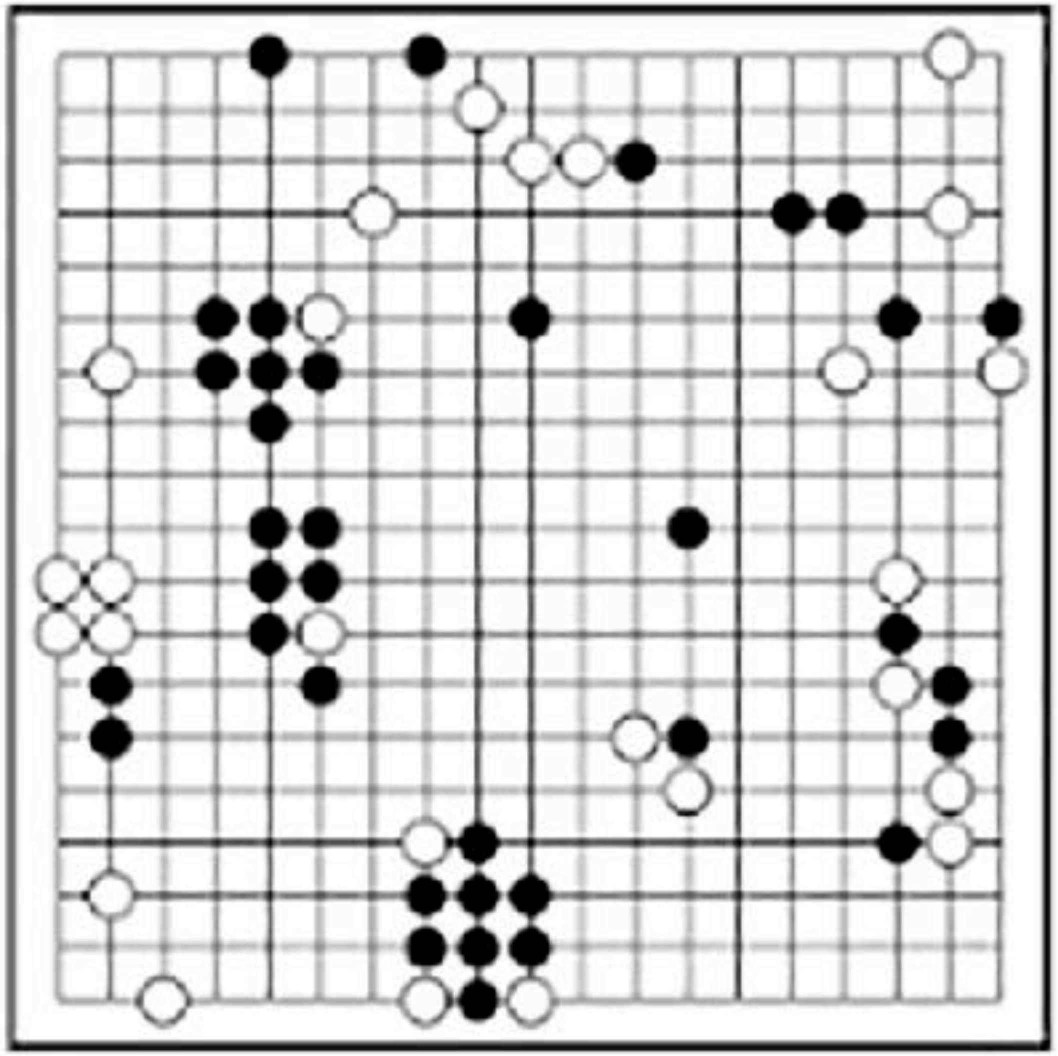
horizontal view

sunny



Markov Decision Process

horizontal view of the game of Go



Solving the optimal policy in MDP

idea:

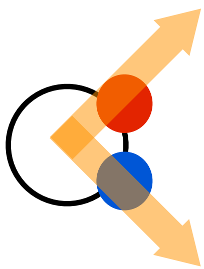
how is the current policy policy evaluation

improve the current policy policy improvement

Policy evaluation

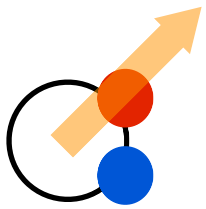
Q: what is the total reward of a policy?

state value function



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

state-action value function



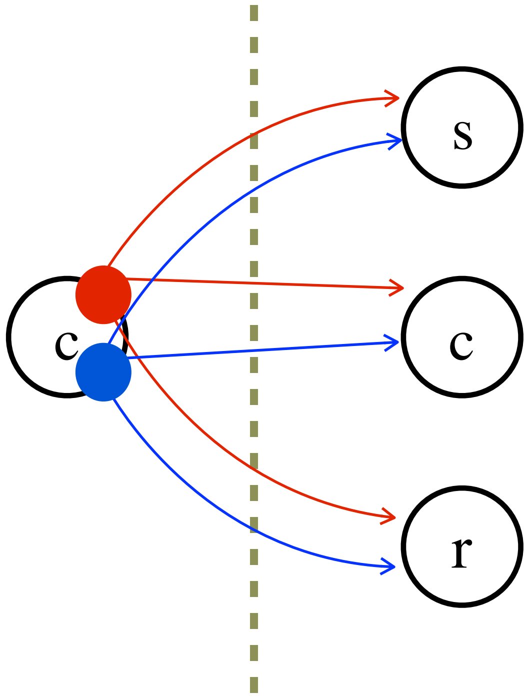
$$Q^\pi(s,a) = E[\sum_{t=1}^{T} r_t | s, a] = \sum_{s'} P(s'|s,a)\big(R(s,a,s') + V^\pi(s')\big)$$

consequently,

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

Policy evaluation

Q: what is the total reward of a policy?

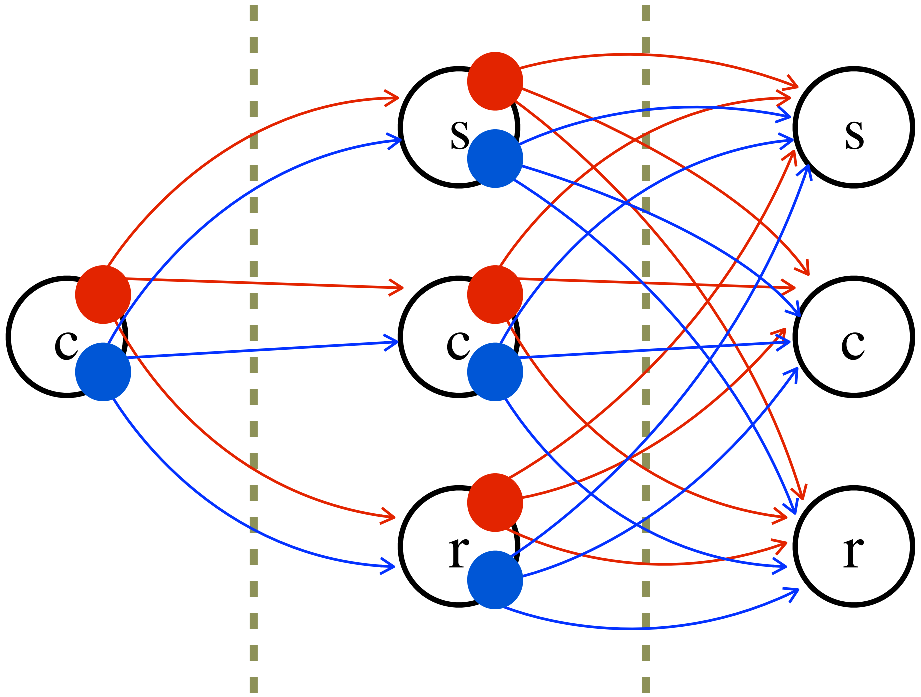


$$V^\pi(s) = \sum_a \pi(a|s) \sum_{s'} P(s'|s,a) R(s,a,s')$$

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

Policy evaluation

Q: what is the total reward of a policy?

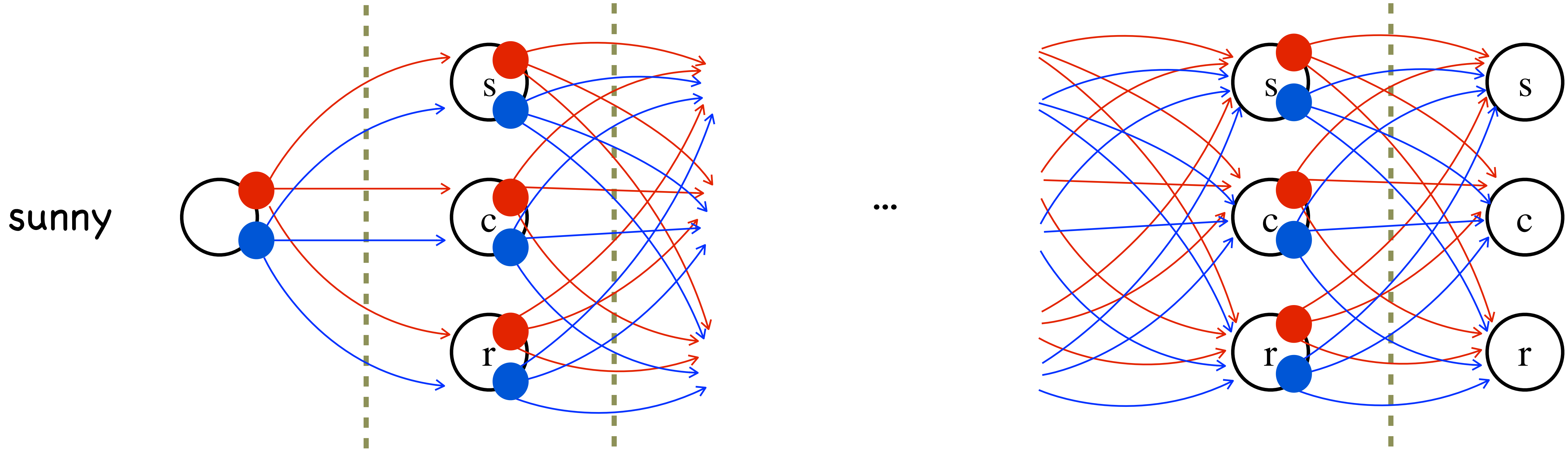


$$V^\pi(s) = \sum_a \pi(a|s) \sum_{s'} P(s'|s,a)\big(R(s,a,s') + V^\pi(s')\big)$$

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

Policy evaluation

Q: what is the total reward of a policy?



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

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

Solving the optimal policy in MDP

idea:

how is the current policy — policy evaluation

improve the current policy — policy improvement

policy iteration:

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

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

value iteration:

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

Solving the optimal policy in MDP

the Bellman optimality equation

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

policy improvement:

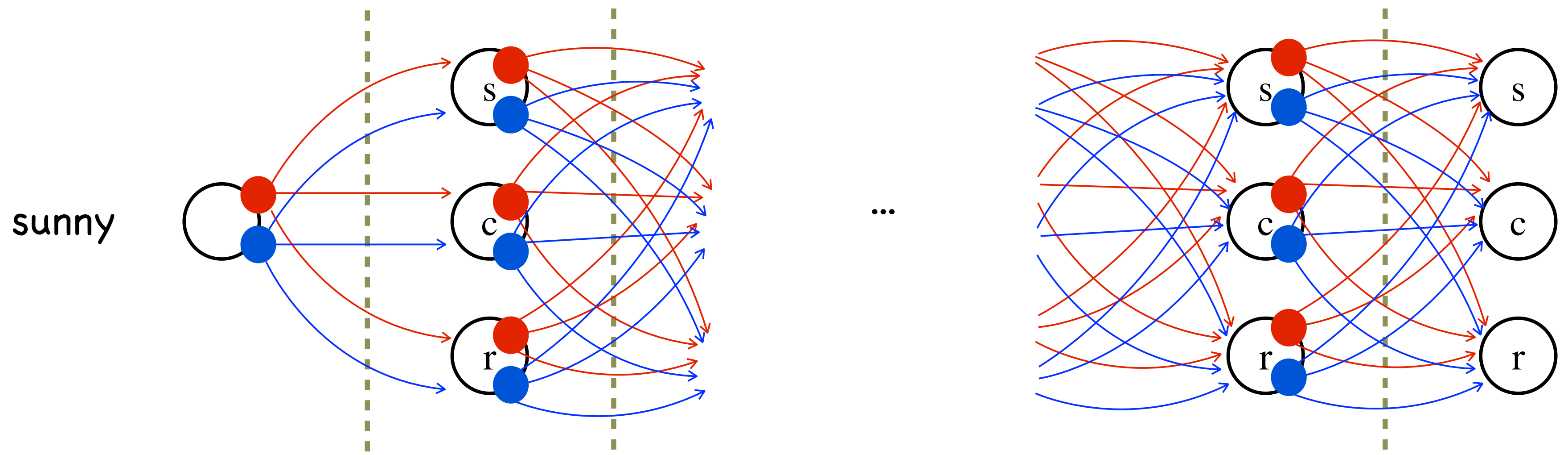
$$V(s) \leftarrow \max_a Q^\pi(s,a)$$

let π' be derived from this update

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

so the policy is improved

Solving the optimal policy in MDP



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, $|S|=2.08\text{x}10^{170}$

[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

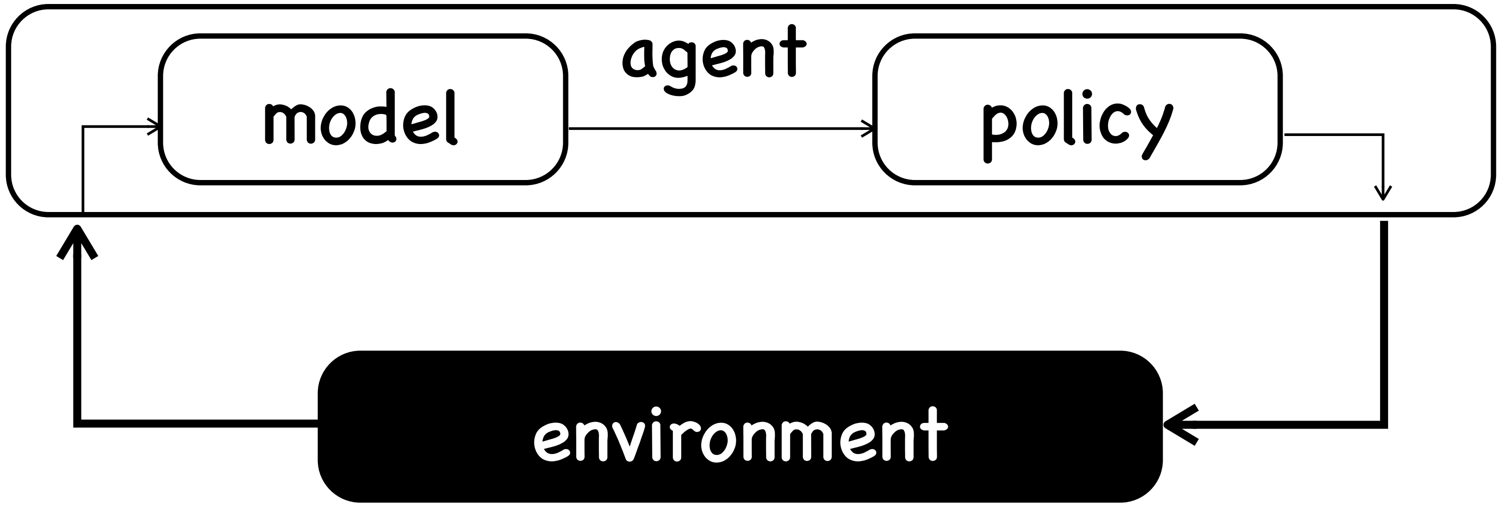
model-based

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 $R(s)$=Rmax, 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

learn policy without knowing MDP

same idea:

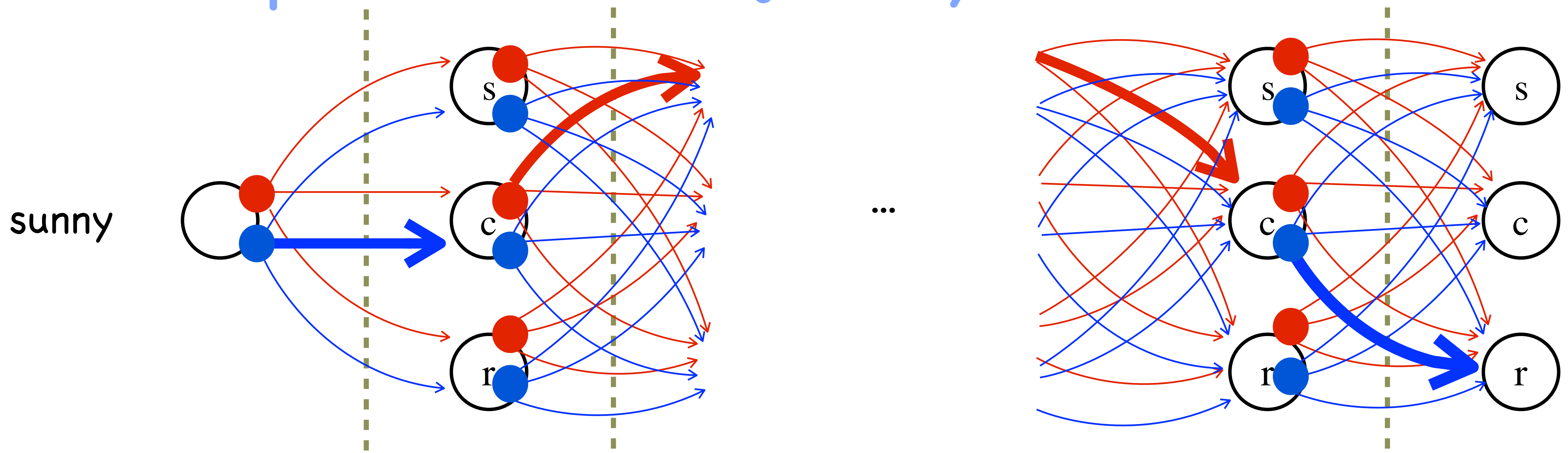
how is the current policy   policy evaluation

improve the current policy   policy improvement

Monte Carlo evaluation

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

same idea:

how is the current policy

Monte-Carlo estimation

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

improve the current policy

policy update

$$\pi(s) = \arg\max_a Q(s,a)$$

Incremental mean

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

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

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

Monte-Carlo evaluation:

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

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

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

    $Q(s_t, a_t) = Q(s_t, a_t) + \alpha(\ R - Q(s_t, a_t)\ )$

  end for

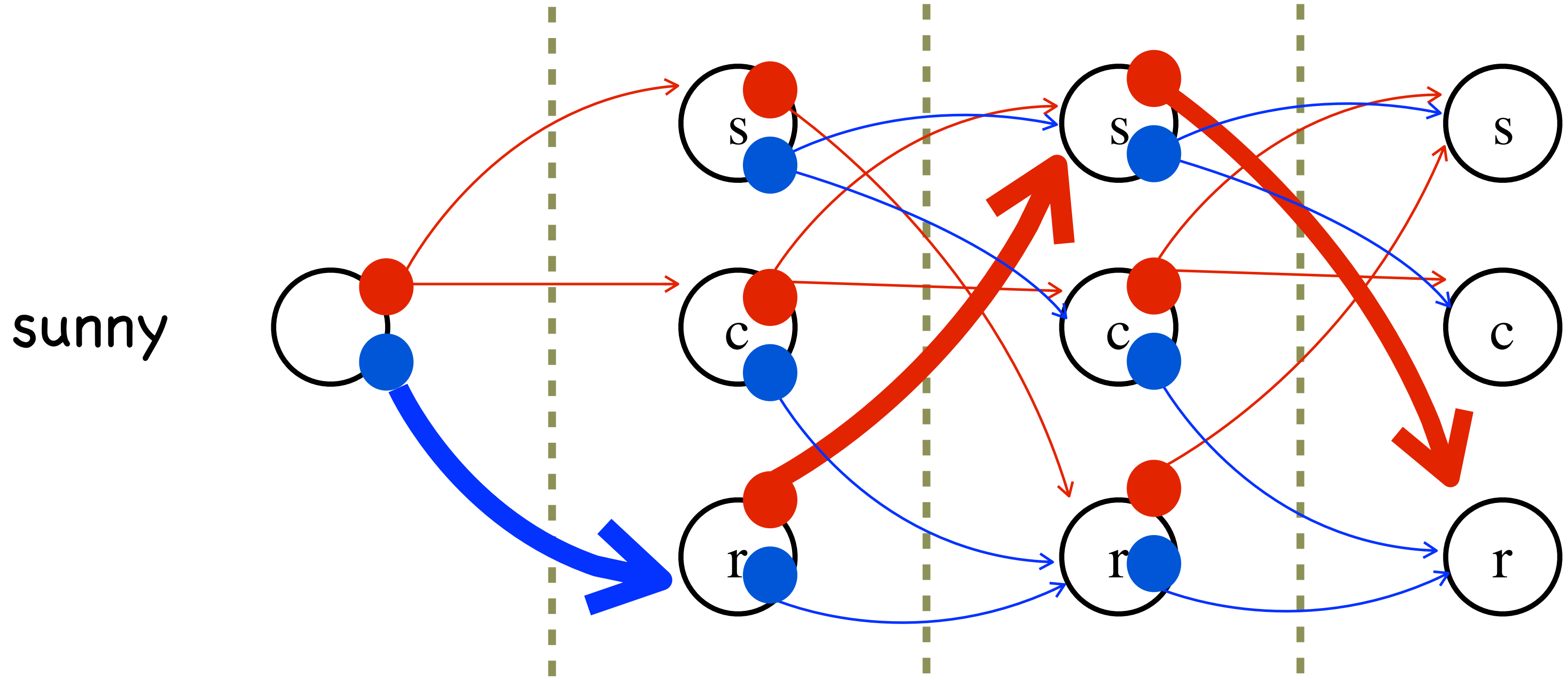
  update policy $\pi(s) = \arg\max_a Q(s, a)$

end for

improvement ?

Monte Carlo RL

problem: what if the policy takes only one path?



cannot improve the policy

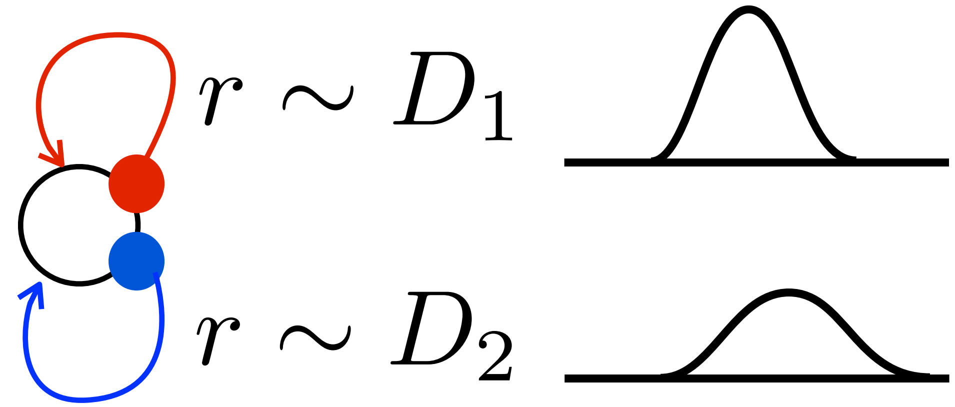
no exploration of the environment

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

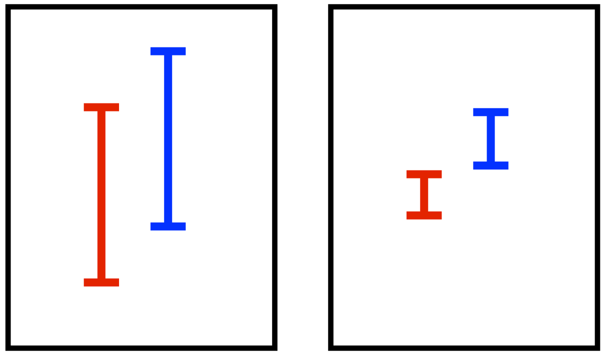
softmax:

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 π_ϵ

for t=0, 1, ..., T-1

$R =$ sum of rewards from t to T

$Q(s_t, a_t) = Q(s_t, a_t) + \alpha(R - Q(s_t, a_t))$

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:

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

$$\Big\downarrow \text{sample from } p \qquad\qquad \Big\downarrow \text{sample from } q$$

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

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 π_ϵ

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)

Temporal-Difference Learning - evaluation

update policy online learn as you go

TD Evaluation

Monte-Carlo update:

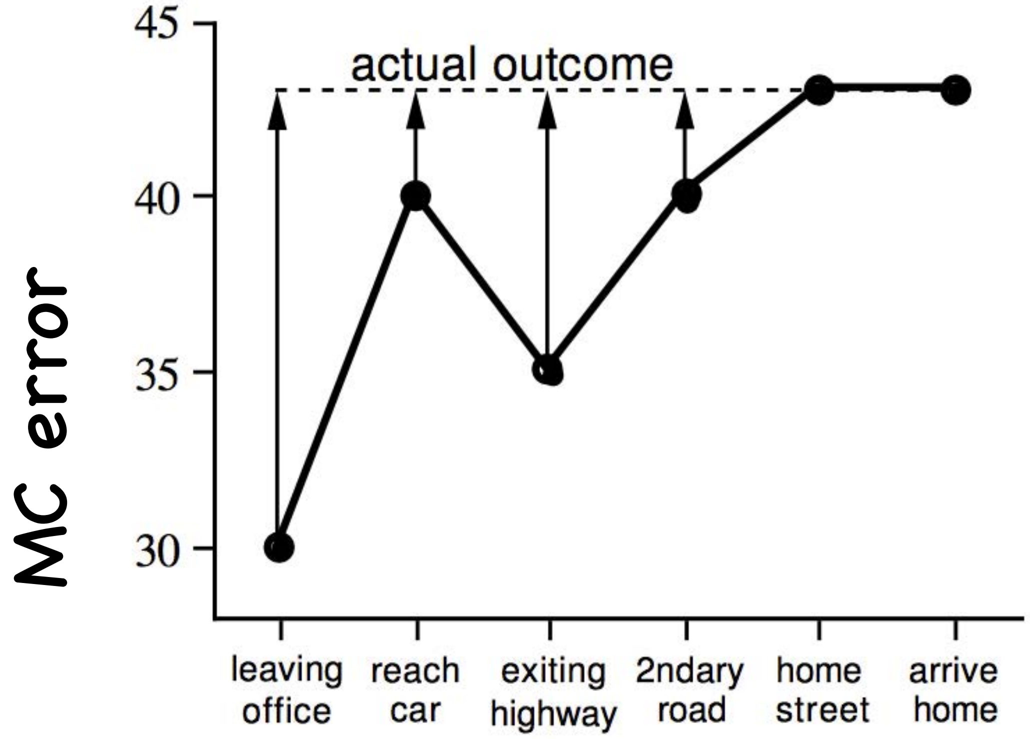
$$Q(s_t, a_t) \Leftarrow Q(s_t, a_t) + \alpha\underbrace{(R - Q(s_t, a_t))}_{\text{MC error}}$$

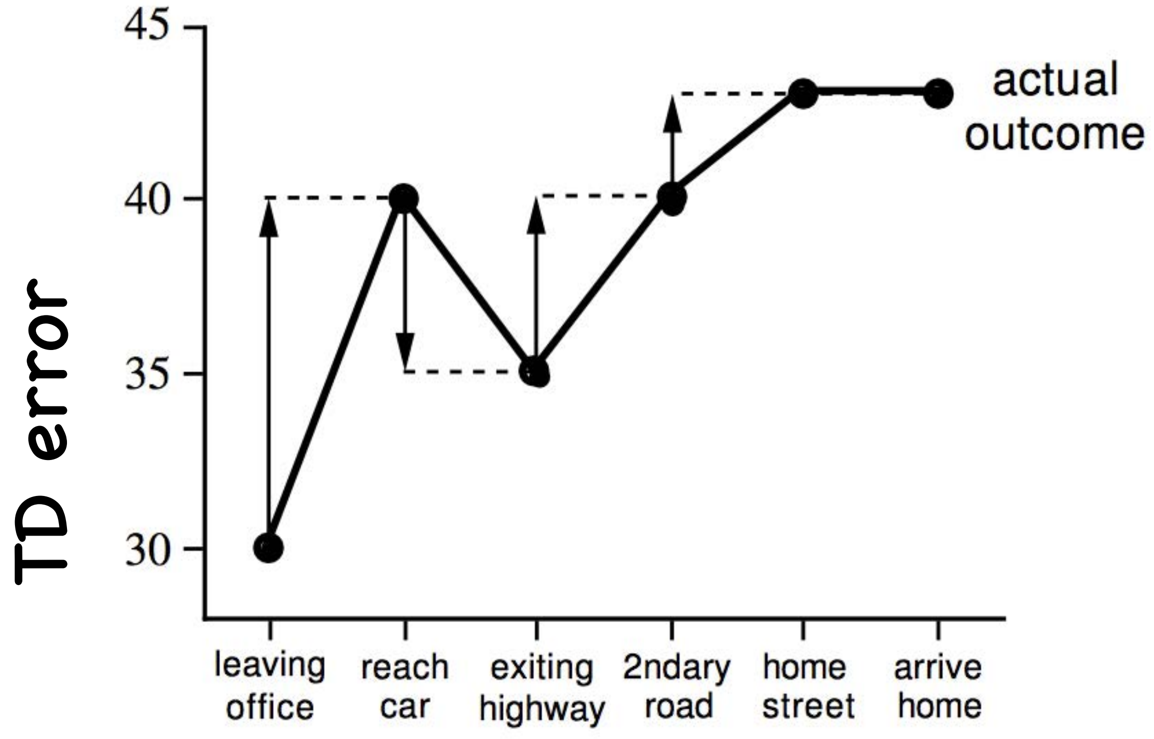
TD update:

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

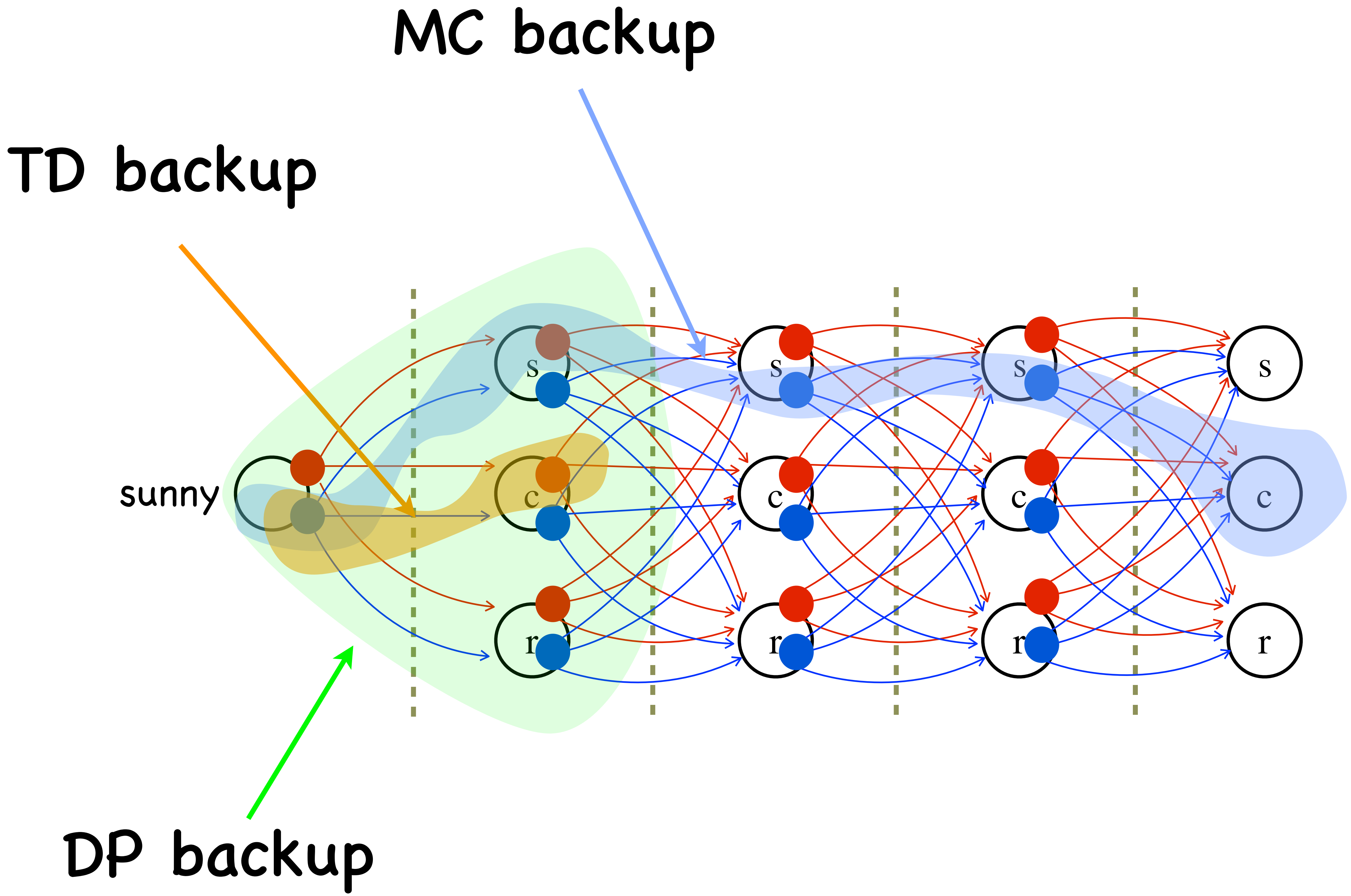
Temporal-Difference Learning - example

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



SARSA

On-policy TD control

$Q_0 = 0$, initial state

for i=0, 1, ...

$\quad a = \pi_\epsilon(s)$

$\quad s', r =$ do action a

$\quad a' = \pi_\epsilon(s')$

$\quad Q(s,a) \mathrel{+}= \alpha(r + \gamma Q(s',a') - Q(s,a))$

$\quad \pi(s) = \arg\max_a Q(s,a)$

$\quad s = s'$

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

$a' = \pi(s')$

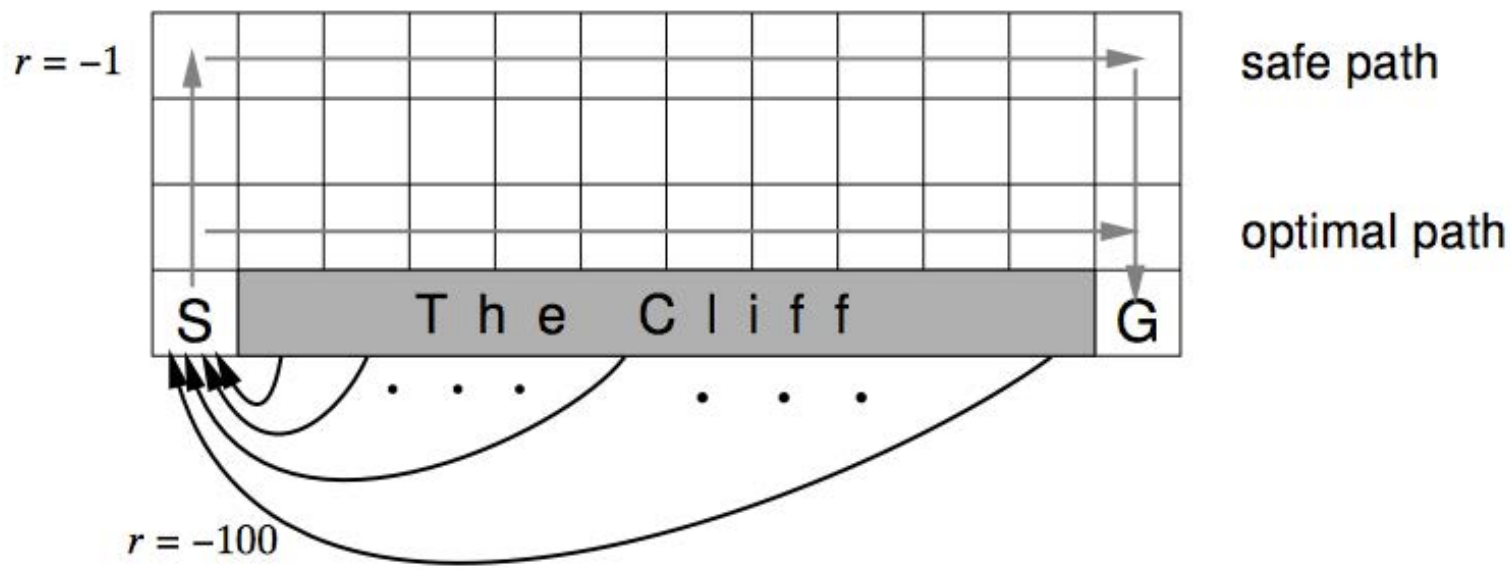
$Q(s,a) \mathrel{+}= \alpha(r + \gamma Q(s', a') - Q(s,a))$

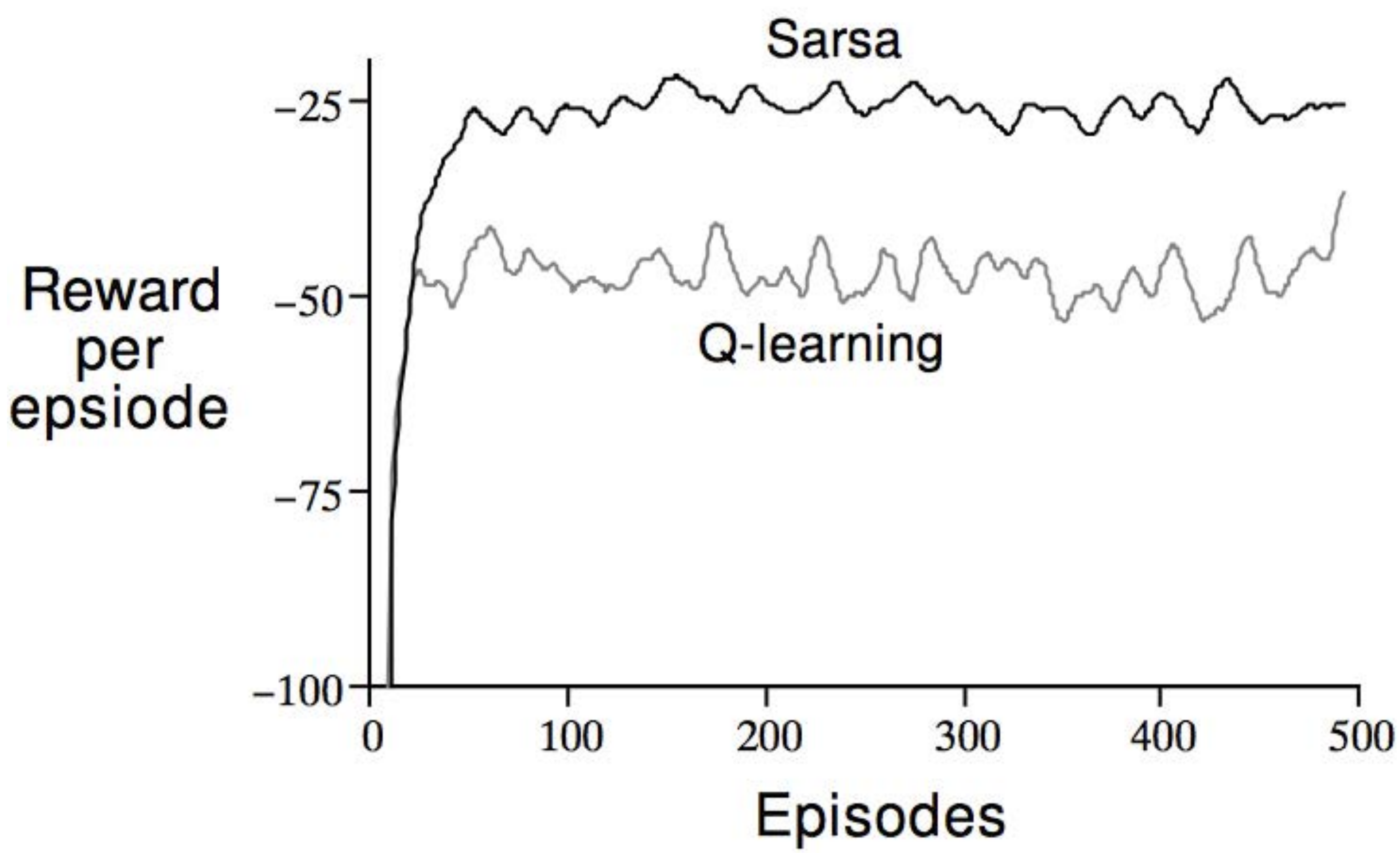
$\pi(s) = \arg\max_a Q(s,a)$

$s = s'$

end for

SARSA v.s. Q-learning

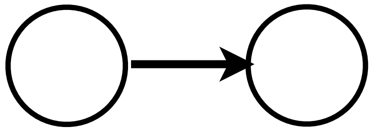




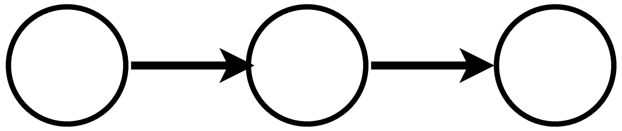
λ-return

in between TD and MC: n-step prediction

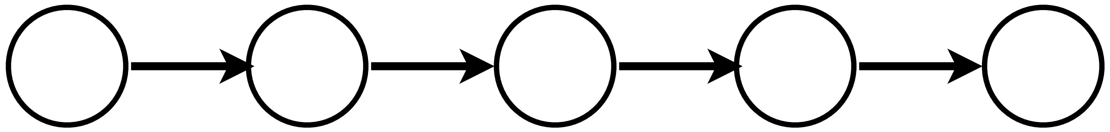
n-step return

TD(1-step) 

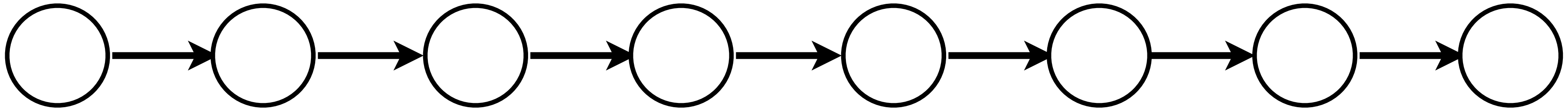
$$R^{(1)} = r_{t+1} + \gamma Q(s_{t+1}, a_{t+1})$$

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 

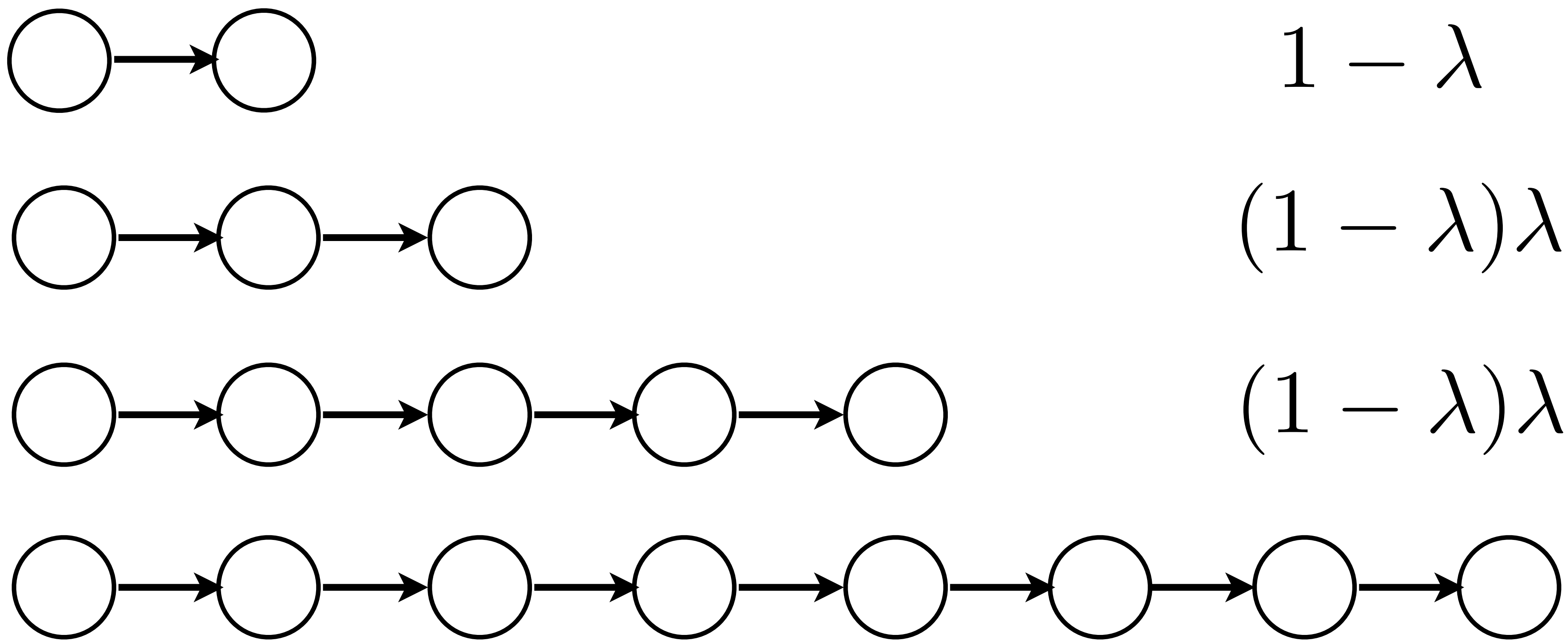
$$R^{(\max)} = \sum_{i=1}^{T} \gamma^{i-1} r_{t+i}$$

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

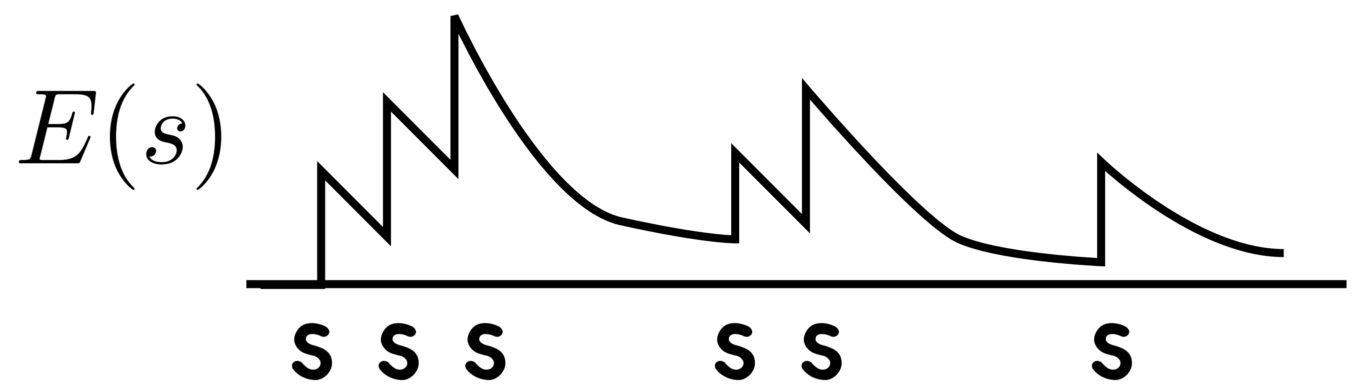
TD(λ): $Q(s_t, a_t) = Q(s_t, a_t) + \alpha(R^\lambda - Q(s_t, a_t))$

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 π_ϵ

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

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

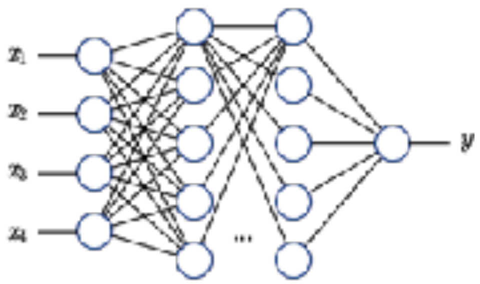
discretization: tide with indicator features

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

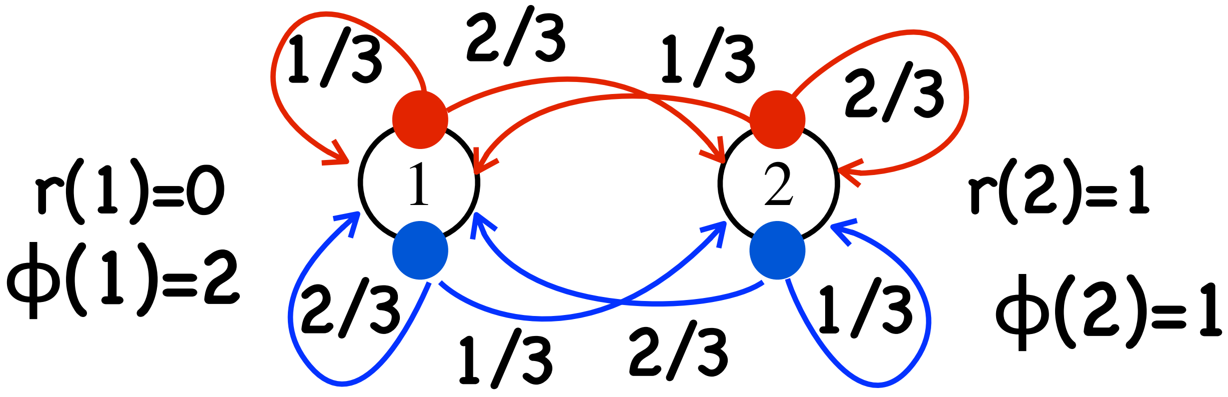
recall the TD update:

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

follow the BP rule to pass the gradient

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 $\neq$ 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 !)

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

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

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

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

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

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

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

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

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)

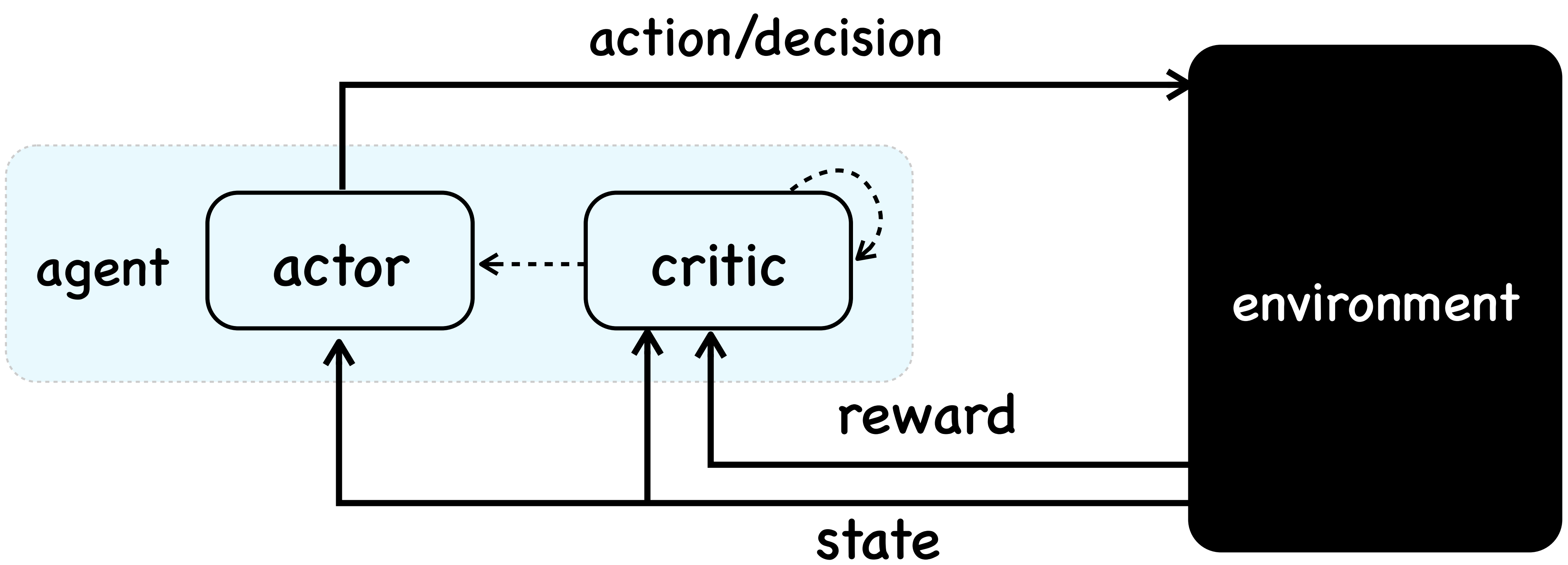
Reduce variance by critic: Actor-Critic

learn policy from trajectories high var. -- actor only

learn value functions low var. -- critic only

combine the two for the good of both:

use critic to stably estimate the gradient



[Grondman, et al. Bartlett. A Survey of Actor-Critic Reinforcement Learning:Standard and Natural Policy Gradients. IEEE Trans. SMC-C, 2012]
[Konda & Tsitsiklis. Actor-Critic Algorithms. NIPS'97]

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

$a' = \pi_\epsilon(s')$

$\delta = r + \gamma Q_w(s', a') - Q_w(s, a)$

$w = w + \alpha\delta\phi(s, a)$

$\theta = \theta + \nabla_\theta \log \pi_\theta(a|s) Q_w(s, a)$

$s = s'$, $a = a'$

end for

Control variance by introducing a bias term

for any bias term b(s)

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

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)

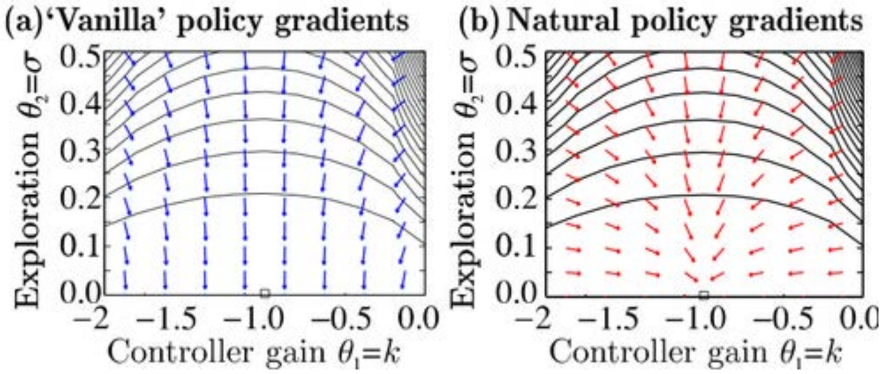
advantage function: $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$

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

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

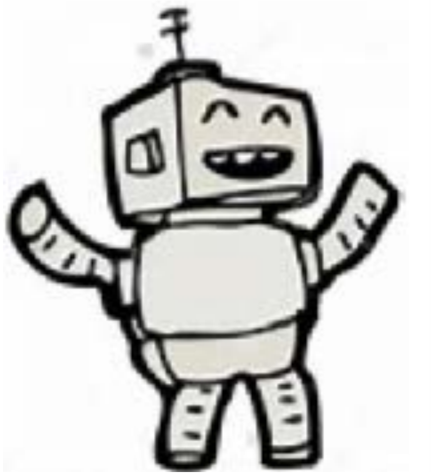
$$\theta \sim \mathcal{N}$$

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

asynchronous gradient update

[Mnih et al. Asynchronous Methods for Deep Reinforcement Learning . ICML'16]

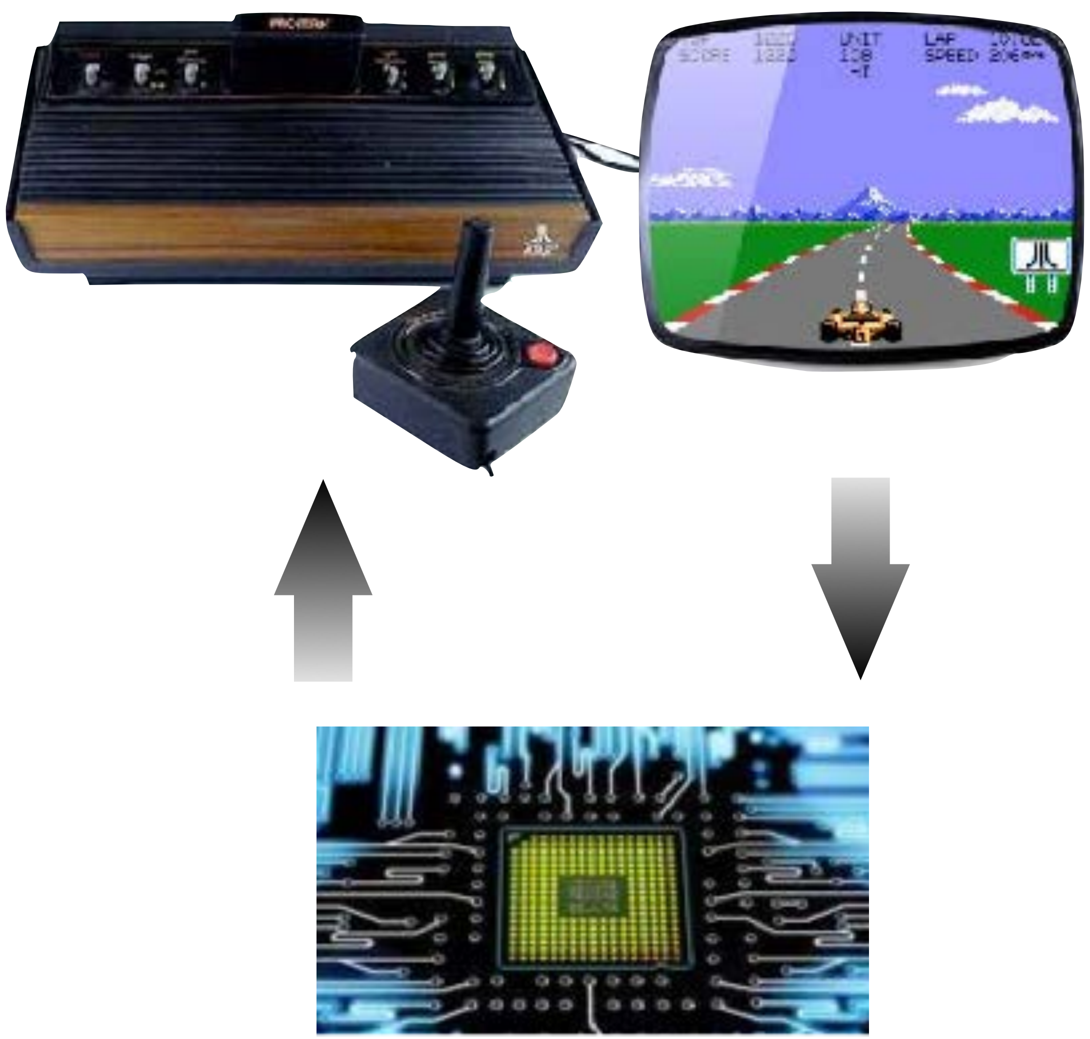
Deep Reinforcement Learning and Games



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]

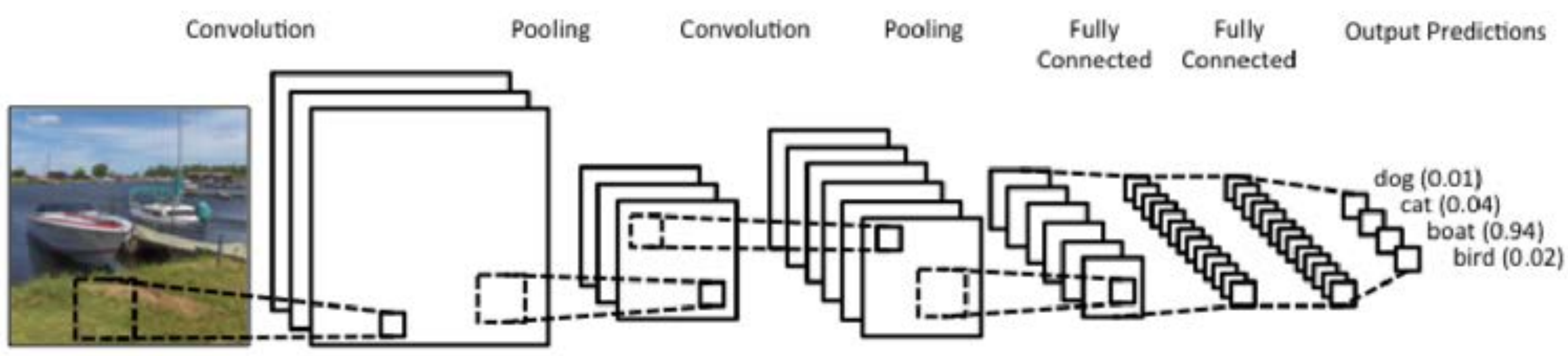


Eye of agent: Deep learning

a powerful architecture for image analysis

differentiable

require a lot of samples to train



Deep reinforcement learning

= deep model + reinforcement learning:

deep model as the function approximation / policy model

How to fit deep neural networks?

stability?

data?

network structure?

…

Deep Q-Network

DQN

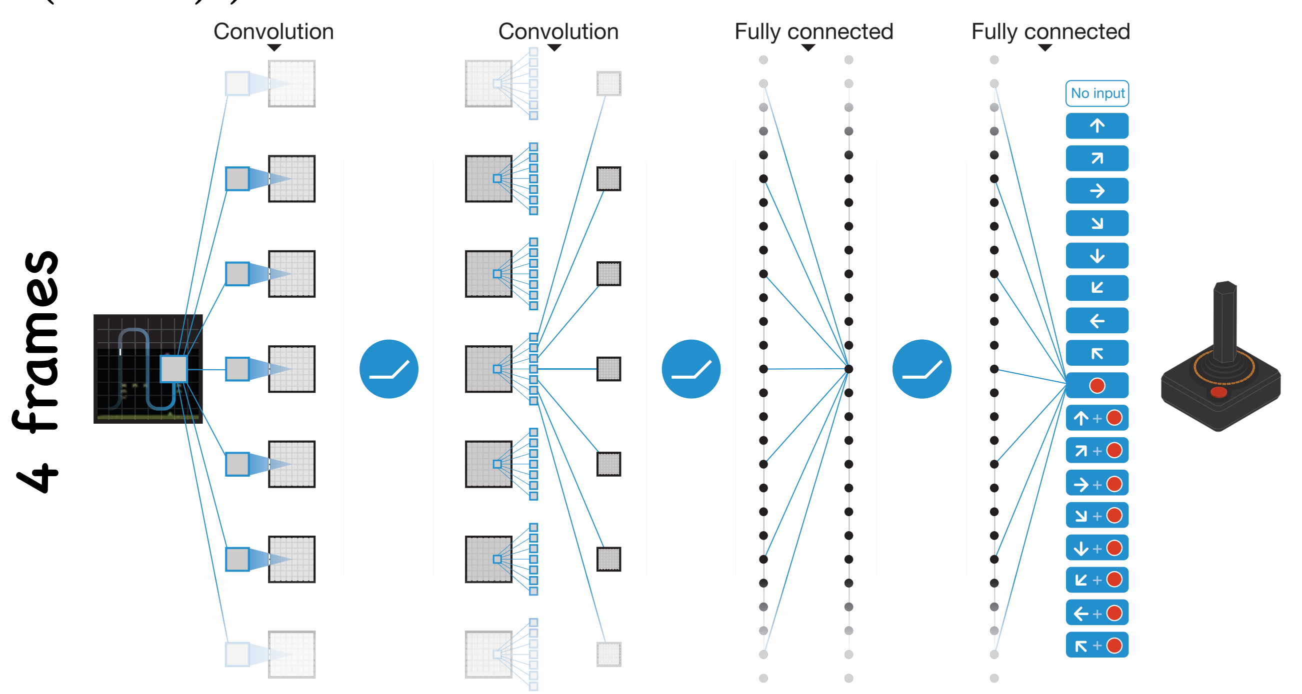
[Mnih *et al*. Human-level control through deep reinforcement learning. Nature, 518(7540): 529-533, 2015]

- 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)

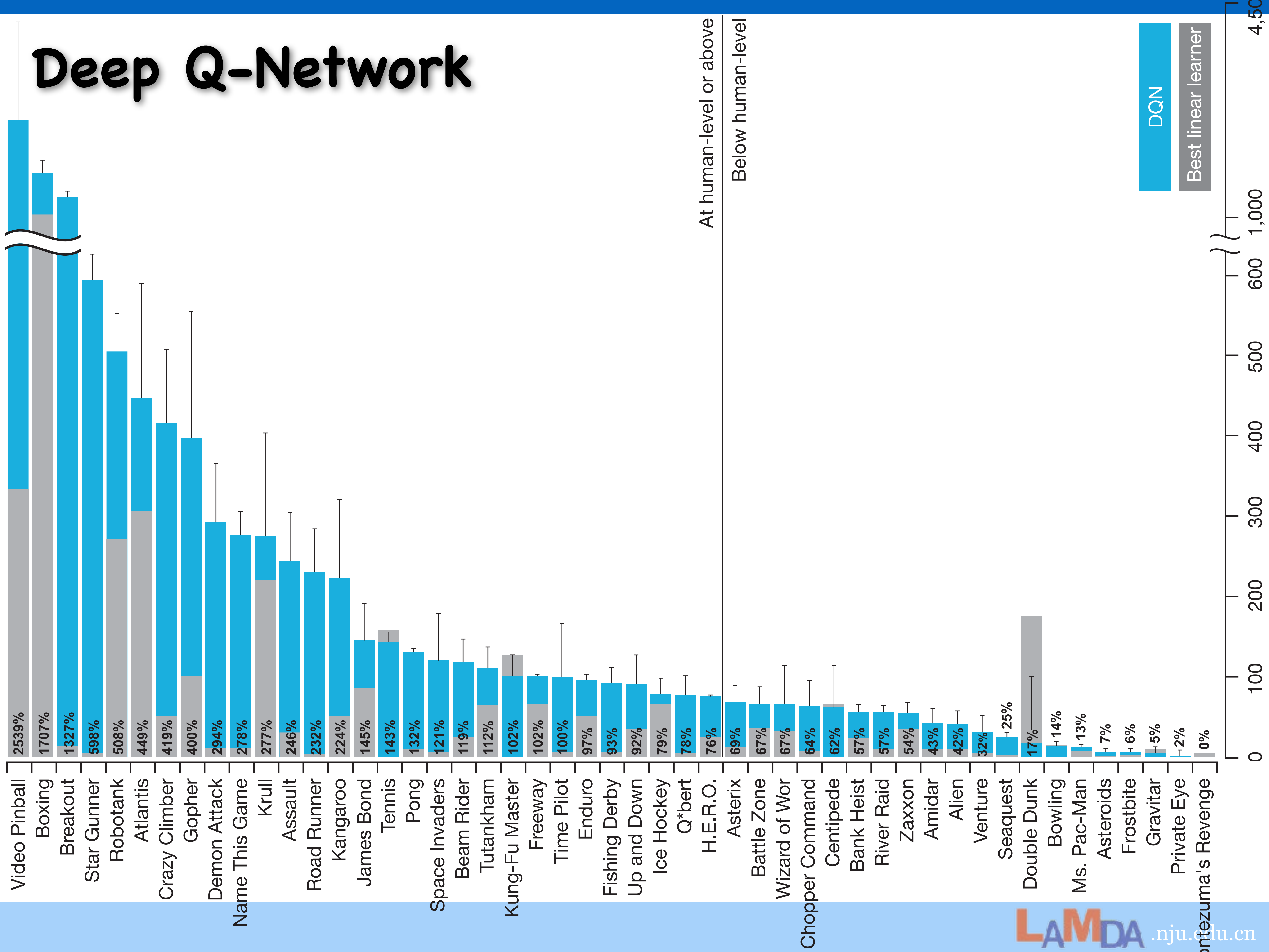
$$\sum (r + \gamma \max_{a'} \tilde{Q}(s', a') - Q_w(s, a))^2$$

DQN on Atari

learn to play from pixels



Deep Q-Network



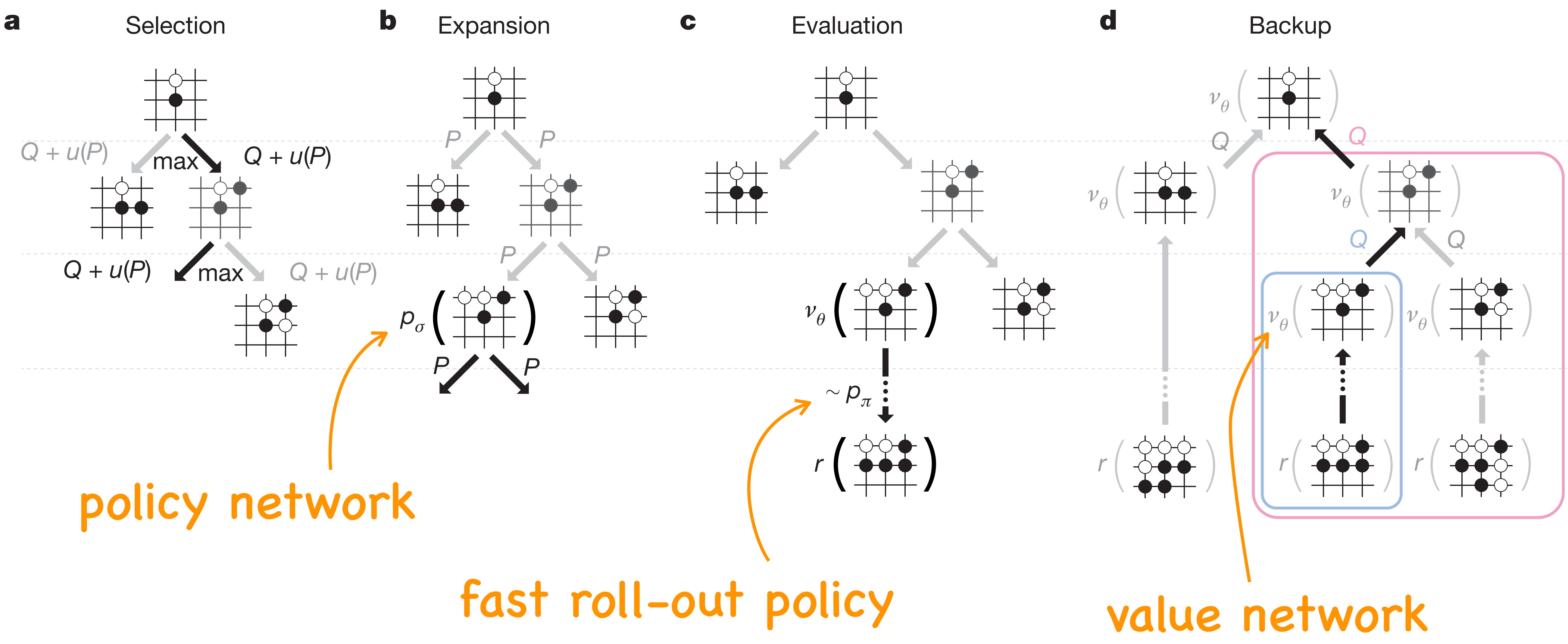
Deep Q-Network

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

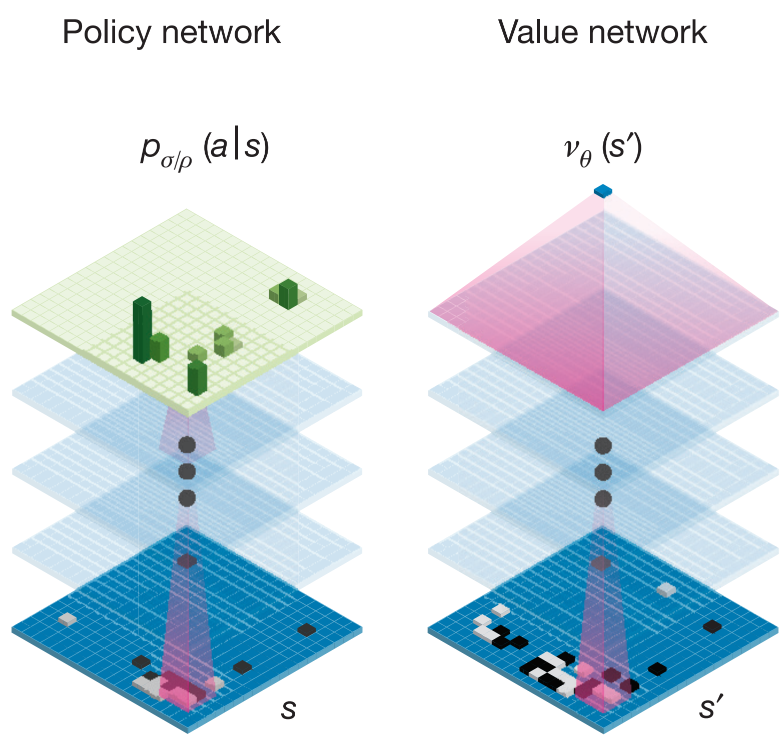
A combination of tree search, deep neural networks and reinforcement learning



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

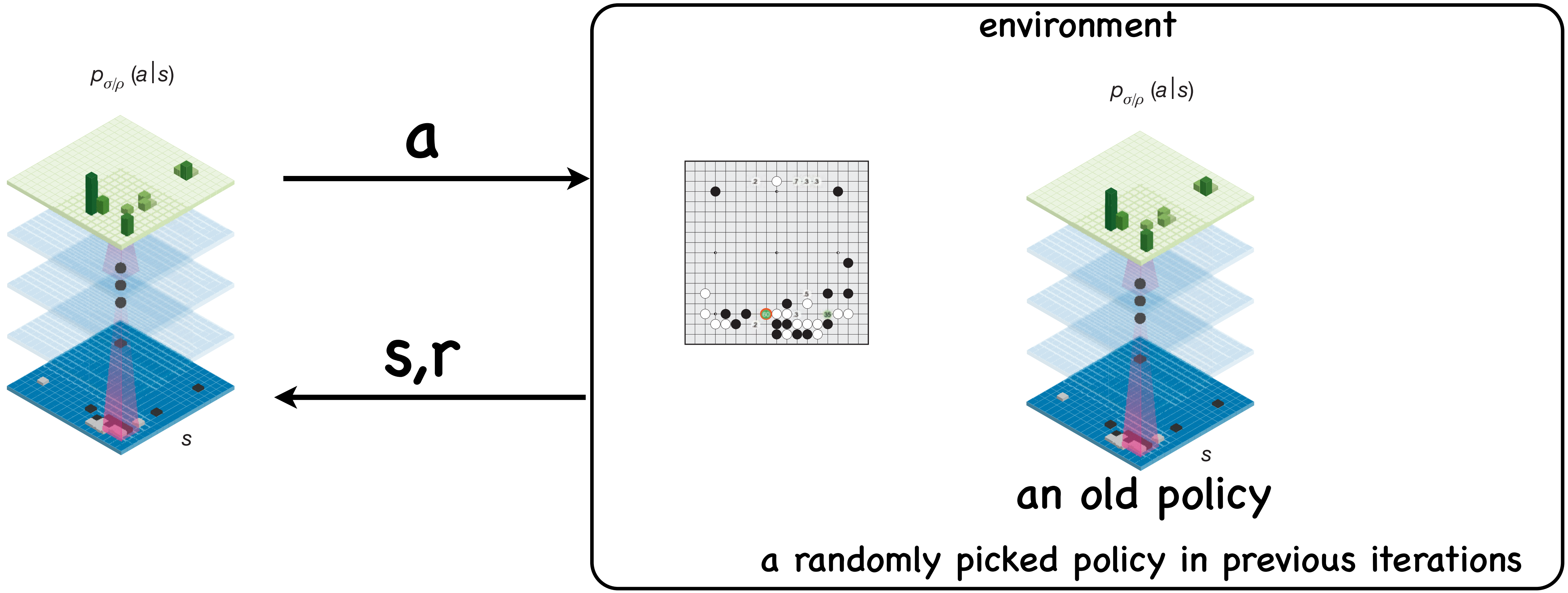
supervised learning from human v.s. human data

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.k.a. self-play

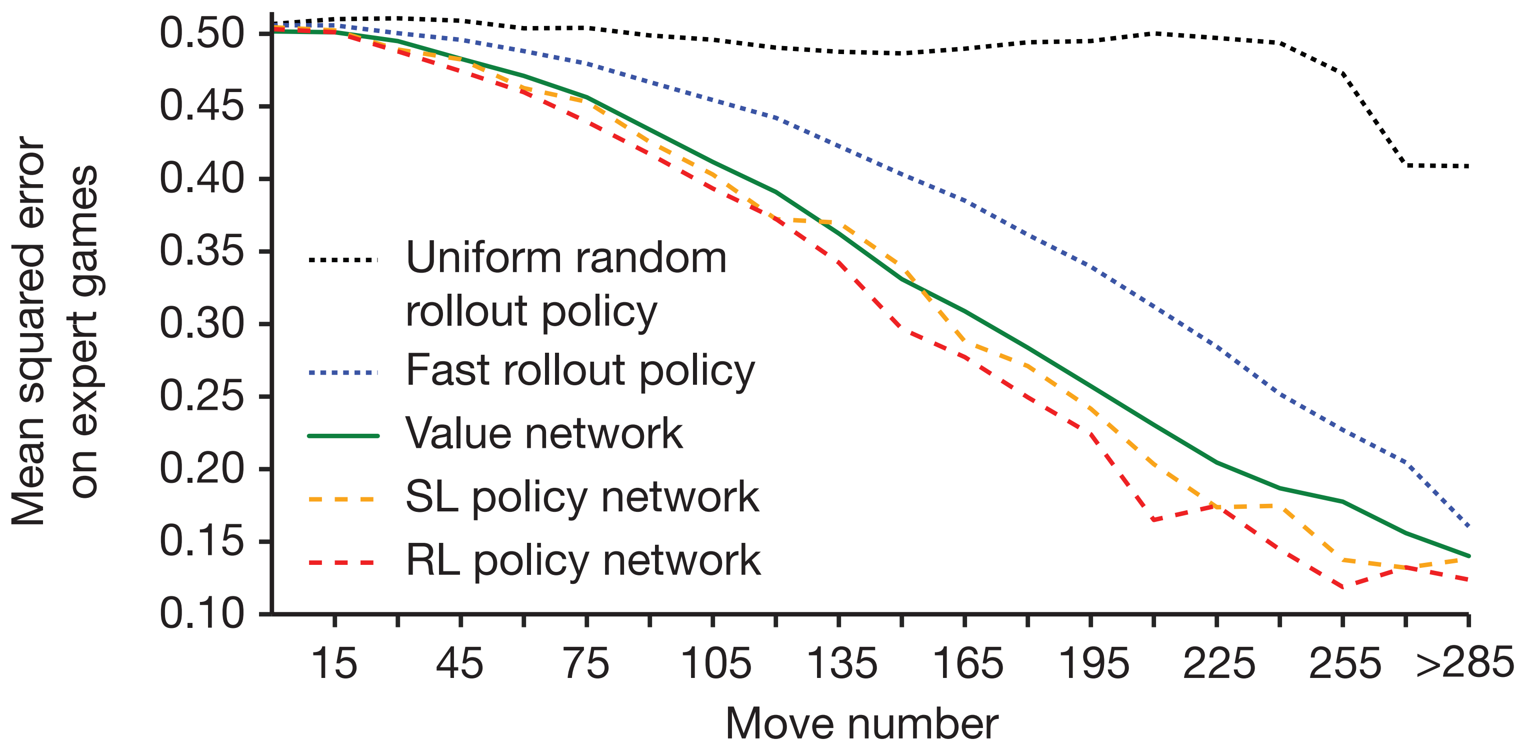
reward:

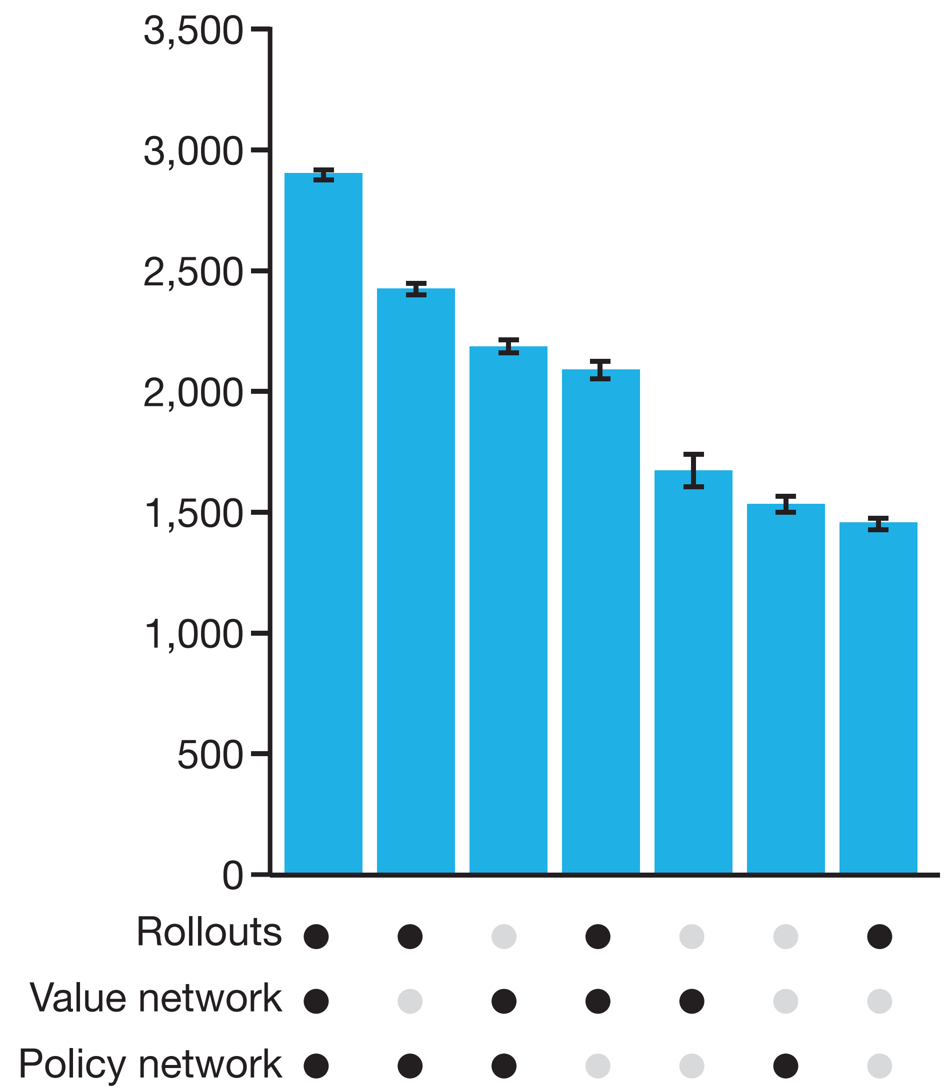
+1 -- win at terminate state

-1 -- loss at terminate state

AlphaGo

value network: supervised learning from RL data





Dueling Network Architectures for Deep Reinforcement Learning

Ziyu Wang ZIYU@GOOGLE.COM
Tom Schaul SCHAUL@GOOGLE.COM
Matteo Hessel MTTHSS@GOOGLE.COM
Hado van Hasselt HADO@GOOGLE.COM
Marc Lanctot LANCTOT@GOOGLE.COM
Nando de Freitas NANDODEFREITAS@GMAIL.COM
Google DeepMind, London, UK

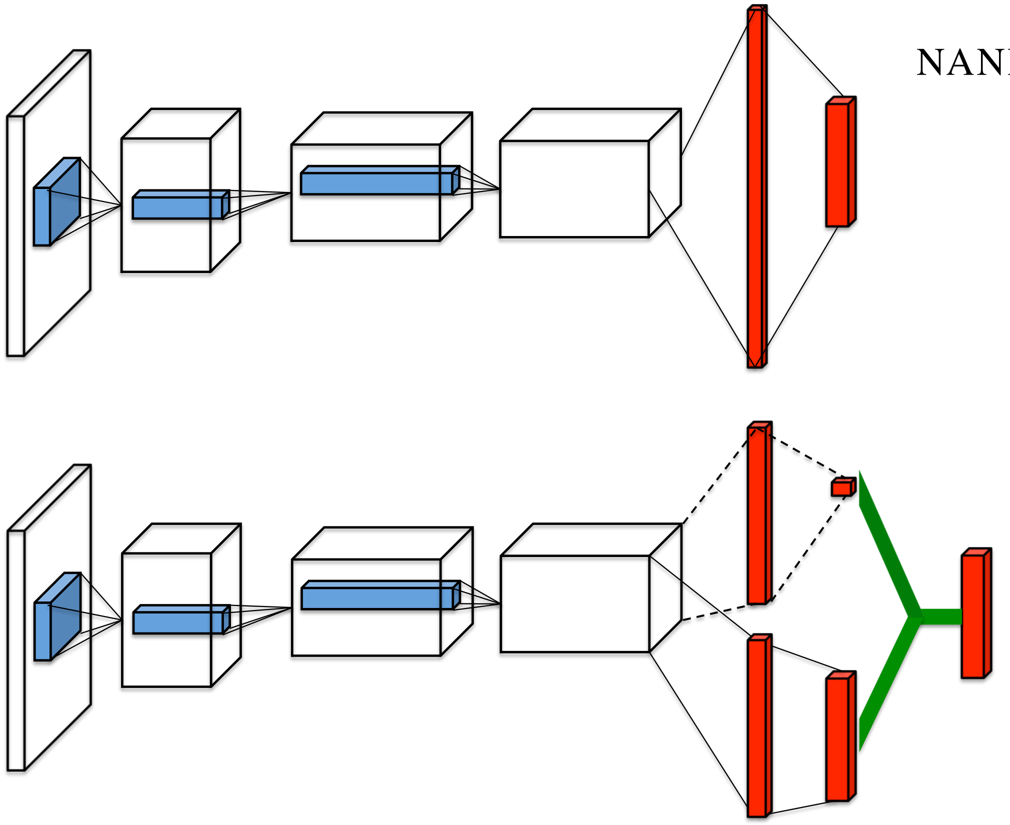


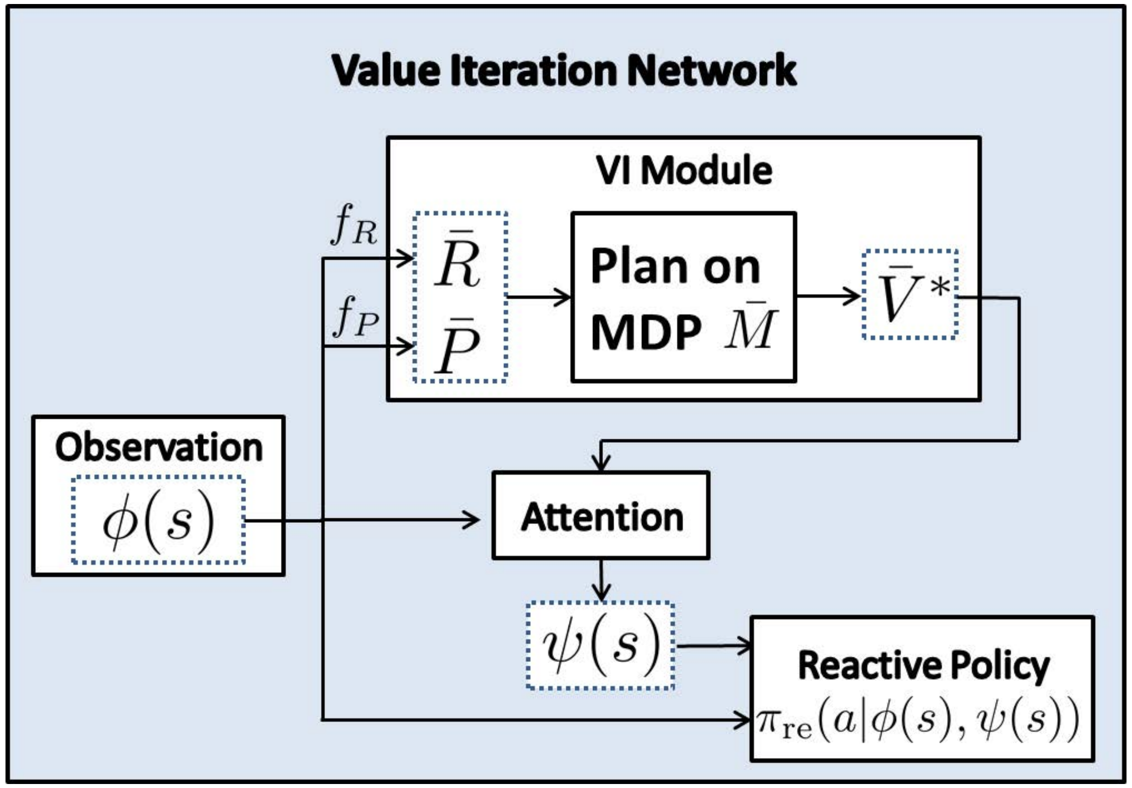
Figure 1. A popular single stream Q-network (**top**) and the dueling Q-network (**bottom**). The dueling network has two streams to separately estimate (scalar) state-value and the advantages for each action; the green output module implements equation (9) to combine them. Both networks output Q-values for each action.

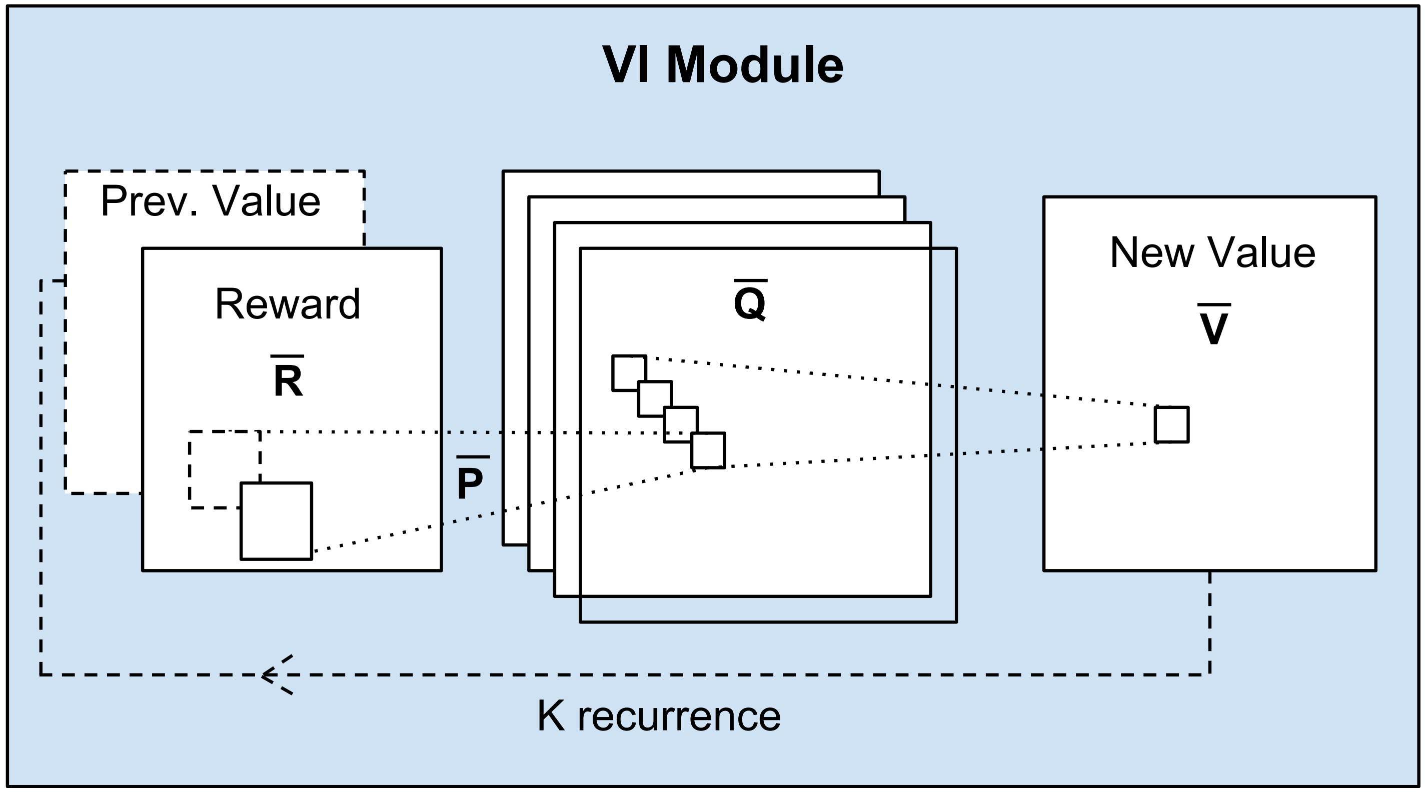
Value Iteration Networks

Aviv Tamar, Yi Wu, Garrett Thomas, Sergey Levine, and Pieter Abbeel

Dept. of Electrical Engineering and Computer Sciences, UC Berkeley

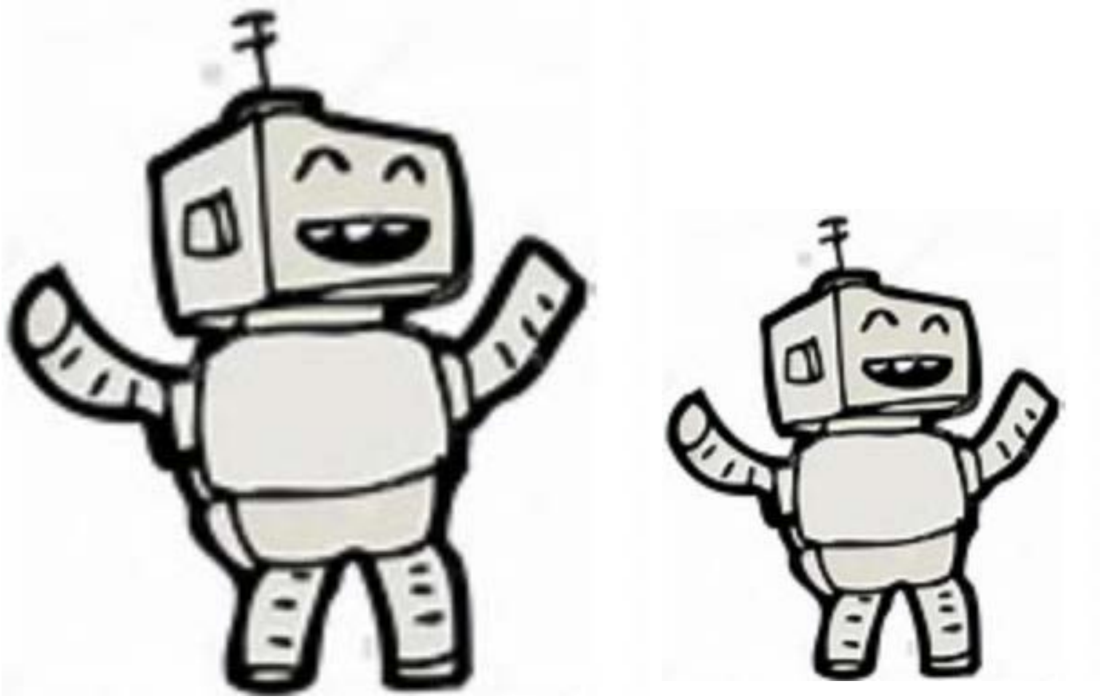
$$V_{t+1}(s) = \max_a \sum_{s'} P(s'|s,a)\big(R(s,a,s') + \gamma V_t(s)\big)$$





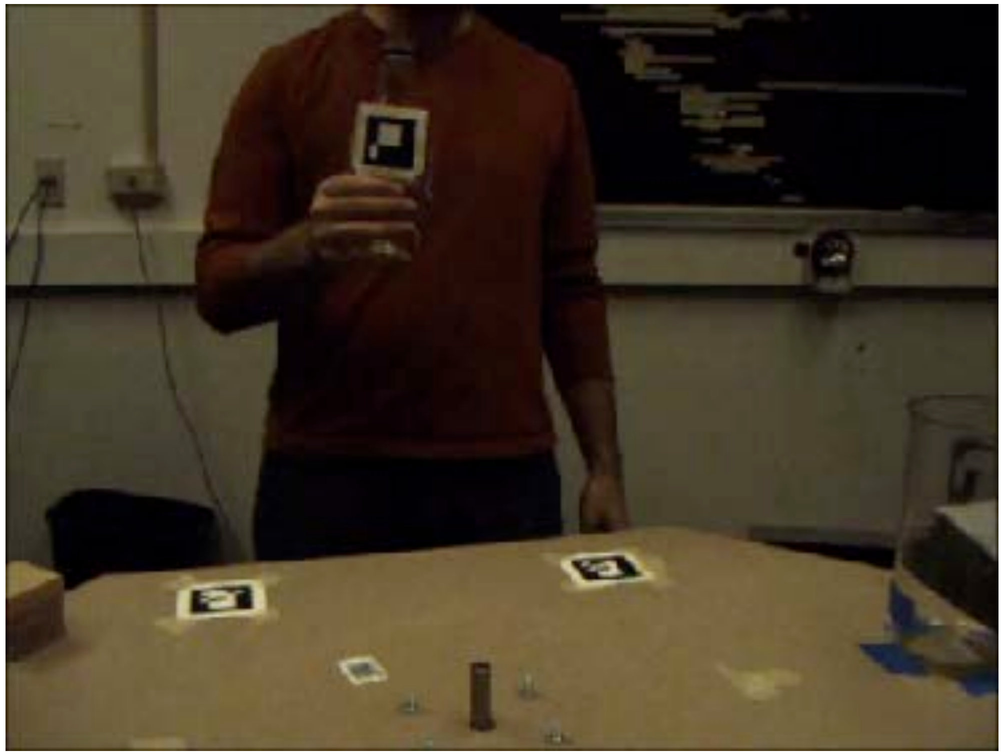
Imitation learning

reinforcement learning with a teacher



Imitation learning

Reinforcement learning is hard due to the delayed feedback



https://www.youtube.com/watch?v=ydnjS__8Ooc

Introducing experts' behavior data:

1. learn faster — immediate feedback
2. learn better — more human like

[Finn et al., ICML'2016]

[Sliver et al., RSS'2008]

[Abbeel et al., IJRR'2010]

Settings

RL problem: $<S, A, R, P>$

Observation: $<S', A', R', P'>$

Simplest: $S=S', A=A', P=P'$ internal data

Hardest: $S \neq S', A \neq A', P \neq P'$ observational data

Can practice: P ?

No — only from demonstration data

Yes — try in the environment

Environment reward accessible: R ?

No — simulate the expert

Yes — maximize the reward

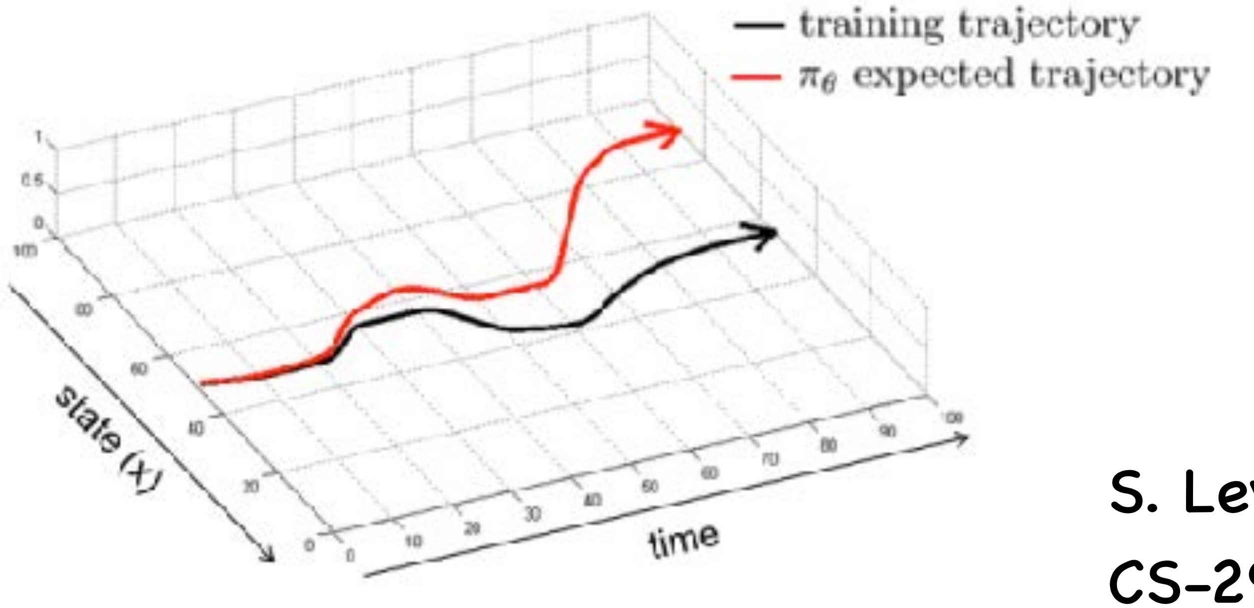
Behavioral Cloning

no practice, no reward

Demonstration: $(s_1, a_1, s_2, a_2, \ldots)$

(s_i, a_i) -> supervised learning

Compounding errors:



S. Levine,
CS-294-112-2

DAgger: Dataset Aggregation

can practice, no reward

collect training data from the learnt policy
ask expert to label the data

Loop:

1. train $\pi_\theta(\boldsymbol{u}_t|\boldsymbol{o}_t)$ from data $D = \{\boldsymbol{o}_1, \boldsymbol{u}_1, \ldots, \boldsymbol{o}_N, \boldsymbol{u}_N\}$
2. run $\pi_\theta(\boldsymbol{u}_t|\boldsymbol{o}_t)$ to get dataset $D_\pi = \{\boldsymbol{o}_1, \ldots, \boldsymbol{o}_M\}$
3. Ask expert to label D_π with actions $\boldsymbol{u}_t$
4. Aggregate: $D \leftarrow D \cup D_\pi$

S. Ross et al.,
AISTATS'2011

Inverse Reinforcement Learning

can practice, no reward

Find a reward function R^* which can explain the expert's behavior

Find R^* such that:

Assume the expert's trajectory is optimal under some reward, so find R^* such that:

$$E[\sum_{t=0}^{\infty} \gamma^t R^*(s_t) \mid \pi^*] \geq E[\sum_{t=0}^{\infty} \gamma^t R^*(s_t) \mid \pi], \forall \pi$$

ill formulation

Inverse Reinforcement Learning

Different reward function formalizations

- Max-margin
  - Feature boosting [Ratliff et al., 2007]
  - Hierarchical formulation [Kolter et al., 2008]
- Feature expectation matching
  - Two player game of feature matching [Syed et al., 2008]
  - Max entropy of feature matching [Ziebart et al., 2008]
- Interpret reward function as parameterization of a policy class
  - Bayesian IRL [Ramachandran et al., 2007]

Brief History for IRL

energy-based models via sampling to estimate the partition function	•Sample-based IRL(Guided Cost Learning) [Finn et al., ICML'2016]
	•Nonlinear IRL with DGP [Wulfmeier et al., NIPS'2015]
	•Bayesian nonparametric feature construction IRL [Choi et al., IJCAI'2014]
parameterized reward function to realize nonlinear IRL	•Nonlinear IRL with GP [Levine et al., NIPS'2011]
state feature construction to realize nonlinear IRL	•Feature construction IRL [Levine et al., NIPS'2010]
	•Maximum entropy IRL [Ziebart et al., AAAI'2008]
dual model to learn the occupancy measure	•Linear programming for Apprenticeship Learning [Syed & Schapire, ICML'2008]
	•Max-margin with boosting [Ratliff et al., NIPS'2007]
	•Max-margin planning [Ratliff et al., AAAI'2006]
	•IRL with exploration policies [Abbeel et al., ICML'2005]
feature expectation match	•Apprenticeship Learning [Abbeel & Ng, ICML'2004]
low-dimensional and discrete state space merely	•First MDP formulation for IRL [Ng & Russell, ICML'2000]
	•1-D inverse optimal control [Kalman et al., ASME1964]

IRL example

Apprenticeship Learning:
find a policy whose performance is close to that of the expert's, on the unknown linear reward function $R^* = w^{*\mathrm{T}}\phi$.

definition for estimator for the expert's feature expectations: $\mu_E = \mu(\pi_E)$
empirical estimation for the estimator: $\hat{\mu}_E = \frac{1}{m}\sum_{i=1}^{m}\sum_{t=0}^{\infty}\gamma^t \phi(s_t^{(i)})$

so we can obtain that if: $\|\mu(\tilde{\pi}) - \mu_E\|_2 \leq \varepsilon$

then:
$$
\begin{aligned}
&|E[\sum_{t=0}^{\infty}\gamma^t R(s_t)\,|\,\pi_E] - E[\sum_{t=0}^{\infty}\gamma^t R(s_t)\,|\,\tilde{\pi}]| \\
&= |w^T\mu(\tilde{\pi}) - w^T\mu_E| \leq \|w\|_2 \|\mu(\tilde{\pi}) - \mu_E\| \\
&\leq 1 \bullet \varepsilon = \varepsilon
\end{aligned}
$$

[Abbeel et al., ICML'2004]

IRL example

So the problem is reduced to finding a policy that induces feature expectations $\mu(\tilde{\pi})$ close to μ_E .

LOOP:

1. randomly pick some policy, compute the feature expectation, and set $i = 1$
2. compute $t^{(i)} = \max_{w:\|w\|_2 \le 1} \min_{j \in \{0..(i-1)\}} w^T (\mu_E - \mu^{(j)})$

and let be the value of $w^{(i)}$ that attains this maximum.

3. if $t^{(i)} < \varepsilon$ then terminate.
4. using RL algorithm to learn the policy under current reward function
5. compute $\mu^{(i)} = \mu(\pi^{(i)})$
6. set $i = i + 1$, and go back to step 2

[Abbeel et al., ICML'2004]

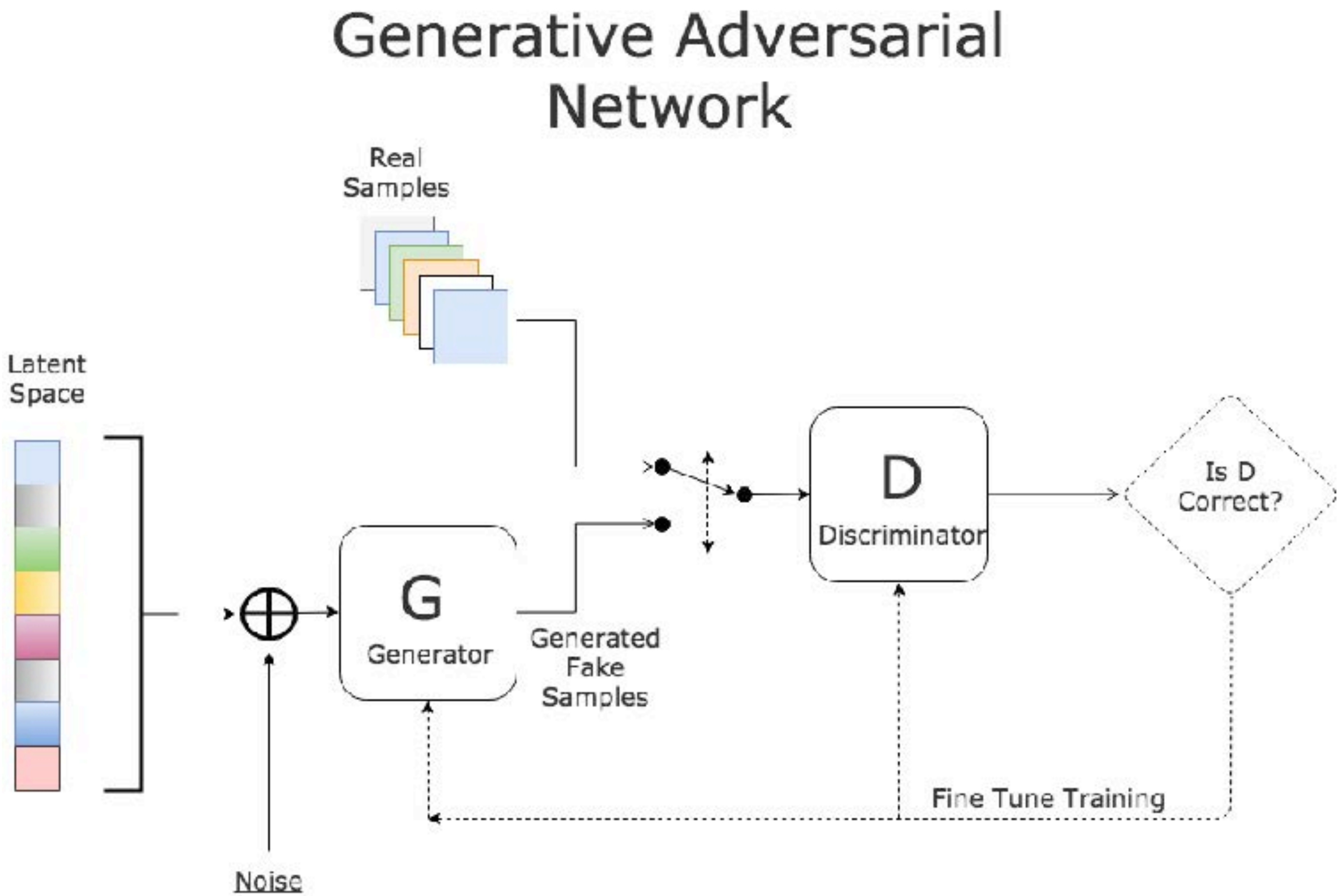
Generative Adversarial Networks

Classical parametric model fitting

e.g. Gaussian distribution: mean and variance

High dimensional data ?

Architecture



[Goodfellow et al., NIPS'2014]

Similarity measures

So some metrics are applied to measure the gap between real data manifold and generated manifold

KL divergence: Not symetrical

Total Variation: $$\delta(\mathrm{P}_r, \mathrm{P}_g) = \sup_{A \in \Sigma} |\mathrm{P}_r(A) - \mathrm{P}_g(A)|$$

JS divergence: $$JS(\mathrm{P}_r, \mathrm{P}_g) = KL(\mathrm{P}_r \| \mathrm{P}_m) + KL(\mathrm{P}_g \| \mathrm{P}_m)$$

Earth-Mover distance: $$W(\mathrm{P}_r, \mathrm{P}_g) = \inf_{\gamma \in \Pi(\mathrm{P}_r, \mathrm{P}_g)} \mathrm{E}_{(x,y)\sim\gamma}[\| x - y \|]$$

...

[M. Arjovsky et al., ICML'2017]

Generative Adversarial Networks

Motivation:
learn the generated data distribution P_g which can be close to the real data distribution P_r.

the goal of GANs is: $\frac{P_r(x)}{P_g(x)+P_r(x)} = \frac{P_g(x)}{P_g(x)+P_r(x)}$

With neural networks to approximate the distribution $D(x) = \frac{P_r(x)}{P_r(x)+P_g(x)}$

Take JS divergence for example,

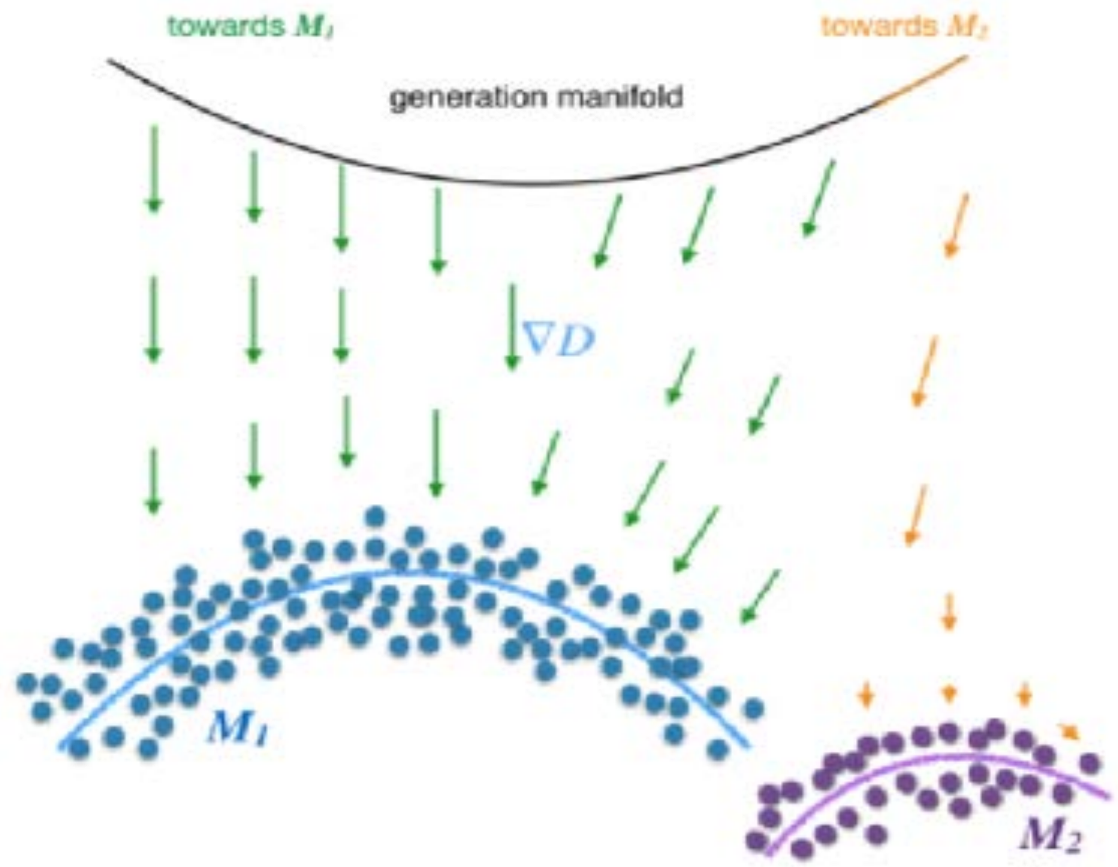
$$JS(P_r, P_g) = \int \log\left(\frac{P_r(x)}{(P_r(x)+P_g(x))/2}\right) P_r(x)\, d\mu(x)$$
$$+ \int \log\left(\frac{P_r(g(z))}{(P_r(g(z))+P_g(g(z)))/2}\right) P_r(g(z))\, d\mu(g(z))$$

Therefore, we can obtain the objective function for GANs:

$$\mathrm{E}_{x\sim p_r}[\log(D(x))] + \mathrm{E}_{z\sim p_g}[\log(1-G(z))]$$

[Goodfellow et al., NIPS'2014]

Mode Collapse Problem of GANs

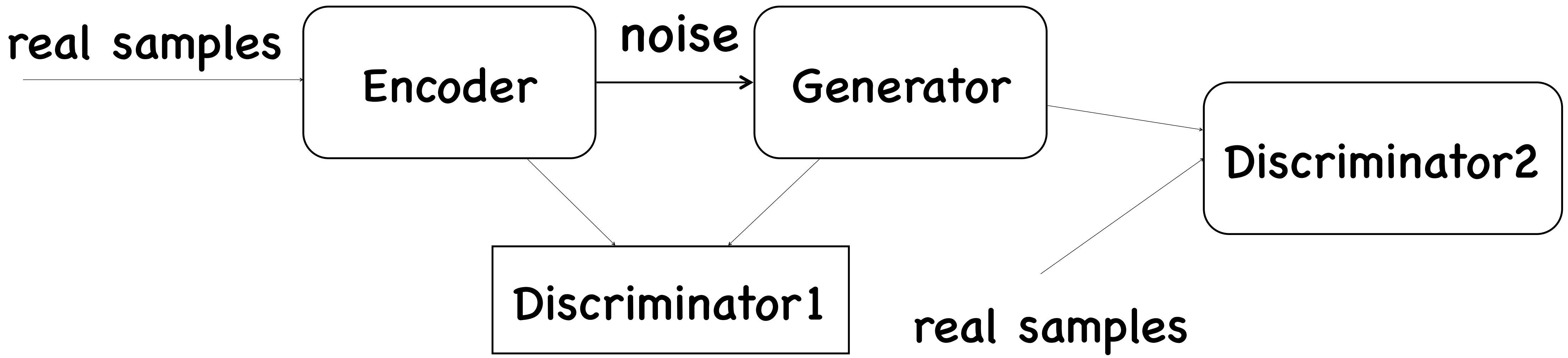


Possible reasons:

1. the generated manifold cannot cover all the example at the beginning
2. the training procedure failed to recover from mode collapse failure

Mode Regularization in GANs

e.g.



Functionals:

variational autoencoder: view the decoder as the generator.

generator: confuse the discriminator with a regularized term.

discriminator: try to distinguish two distributions, while discriminator 1 is used to train the variational autoencoder, and discriminator 2 is used to train the GAN procedure

[Che et al., ICLR'2016]

Generative Adversarial Imitation Learning (GAIL)

GAIL realize the procedure for inverse reinforcement learning by finding a saddle point (π, D) of the expression.

$$\mathrm{E}_{\pi}[\log(D(s,a))] + \mathrm{E}_{\pi_\theta}[\log(1 - D(s,a))] - \lambda H(\pi)$$

LOOP:

1. Update the discriminator with the gradient
$$\hat{\mathrm{E}}_{\tau_i}[\nabla_w \log(D_w(s,a))] + \hat{\mathrm{E}}_{\tau_E}[\nabla_w \log(1 - D_w(s,a))]$$
2. Take a policy step with reward function
$$r(s,a) = \log(D(s,a))$$

[Ho et al., NIPS'2016]

Connections between IRL & GAIL

Inverse Reinforcement Learn

$$\max_{c \in C}(\min_{\pi \in \Pi} -H(\pi) + \mathrm{E}_{\pi}[c(s,a)]) - \mathrm{E}_{\pi_E}[c(s,a)]$$

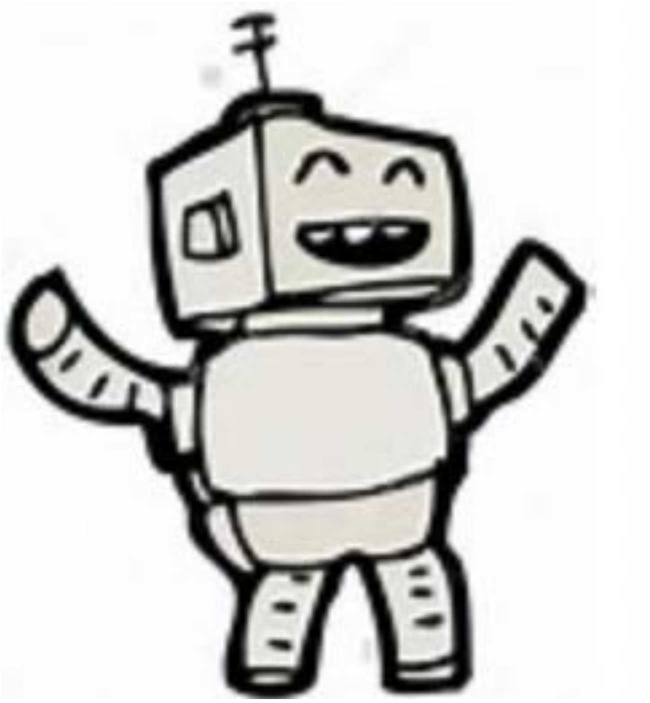
$$H(\pi) = \mathrm{E}_{\pi}[-\log(\pi(a \mid s))]$$

Generative Adversarial Networks:

$$\min_{G} \max_{D} \mathrm{E}_{x \sim p_r}[\log(D(x))] + \mathrm{E}_{z \sim p_g}[\log(1 - D(G(z)))]$$

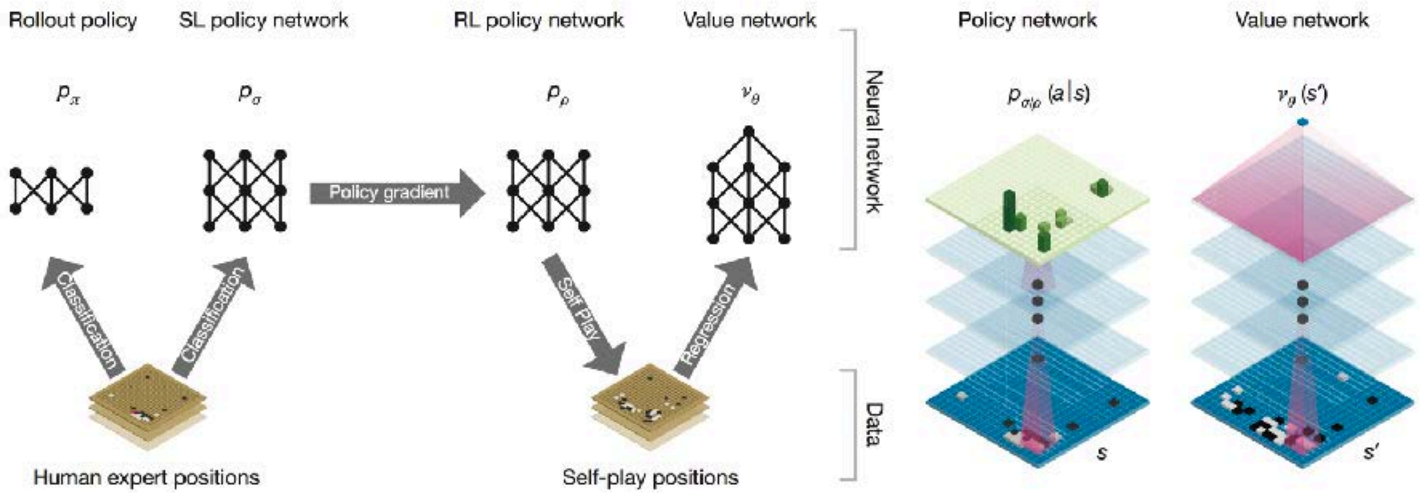
view the discriminator as the reward function for inverse reinforcement learning.

Discussion on the future



Successes of reinforcement learning

Atari games



Game of Go

Mujoco

VisDoom

Dota2

The cost of the success

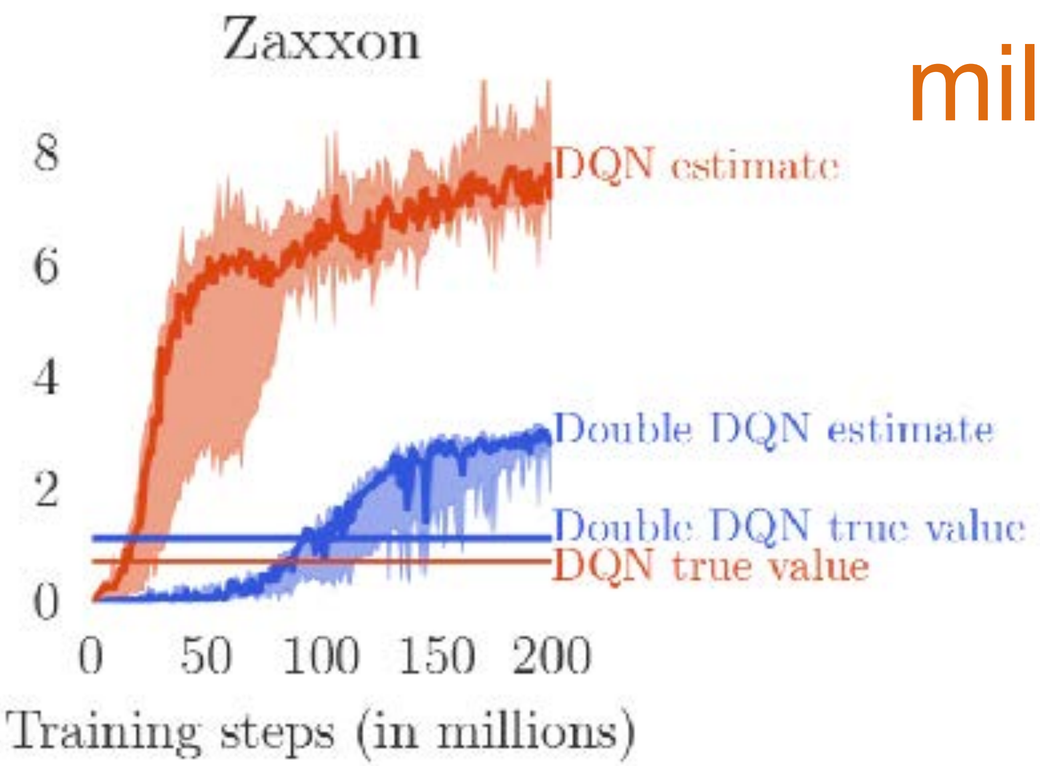


figure from [Hasselt et al., AAAI'16]

millions v.s. tens

Sorta Insightful

In a world where everyone has opinions, one man...also has opinions

Deep Reinforcement Learning Doesn't Work Yet

Feb 14, 2018

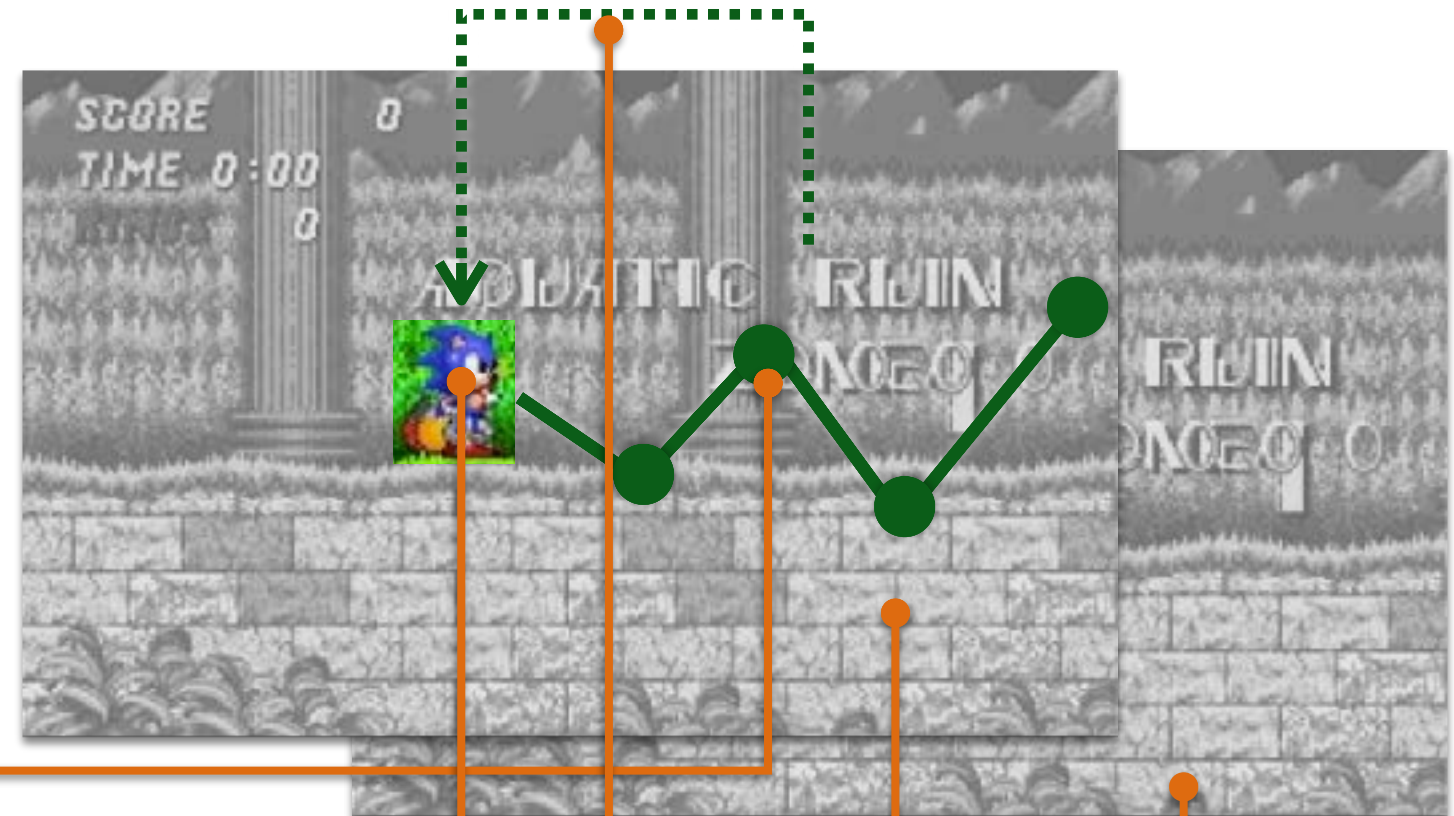
Himanshu Sahni's Blog

Reinforcement Learning never worked, and 'deep' only helped a bit.

FEBRUARY 23, 2018

Why inefficient?

in environments, repeatedly:
- generate trajectories
- update the model



exploration

optimization

environment modeling

experience transfer

abstraction

1 — Exploration

generate new and hopefully better trajectories

classical approach — action space noise:

ε-greedy, Gibbs distribution, Gaussian distribution …

may not friendly for policy update

parameter space noise:

e.g. [Plappert et al., ICLR'18] [Fortunato et al., 2018]

could be more efficient

but still memoryless and blind

curiosity-driven exploration:

easy in simple settings (a few discrete states and actions)

difficult in real state/action spaces

e.g., Intrinsic Curiosity Module [Pathak et al., ICML'17]

The OpenAI Retro Contest

- pixel input
- offline training / online test environments are different
- 1 million steps retrain in test environments
- relatively large environment

our team →

RANK	TEAM	SCORE
#1	Dharmaraja	4692
#2	mistake	4446
#3	aborg	4430
#4	whatever	4274
#5	Students of Plato	4269
	Joint PPO baseline	4070
	Joint Rainbow baseline	3843
	Rainbow baseline	3498

246

177

The OpenAI Retro Contest

curiosity-driven exploration is essential !

we compare images to generate the intrinsic reward

red: early trajectories
blue: later trajectories

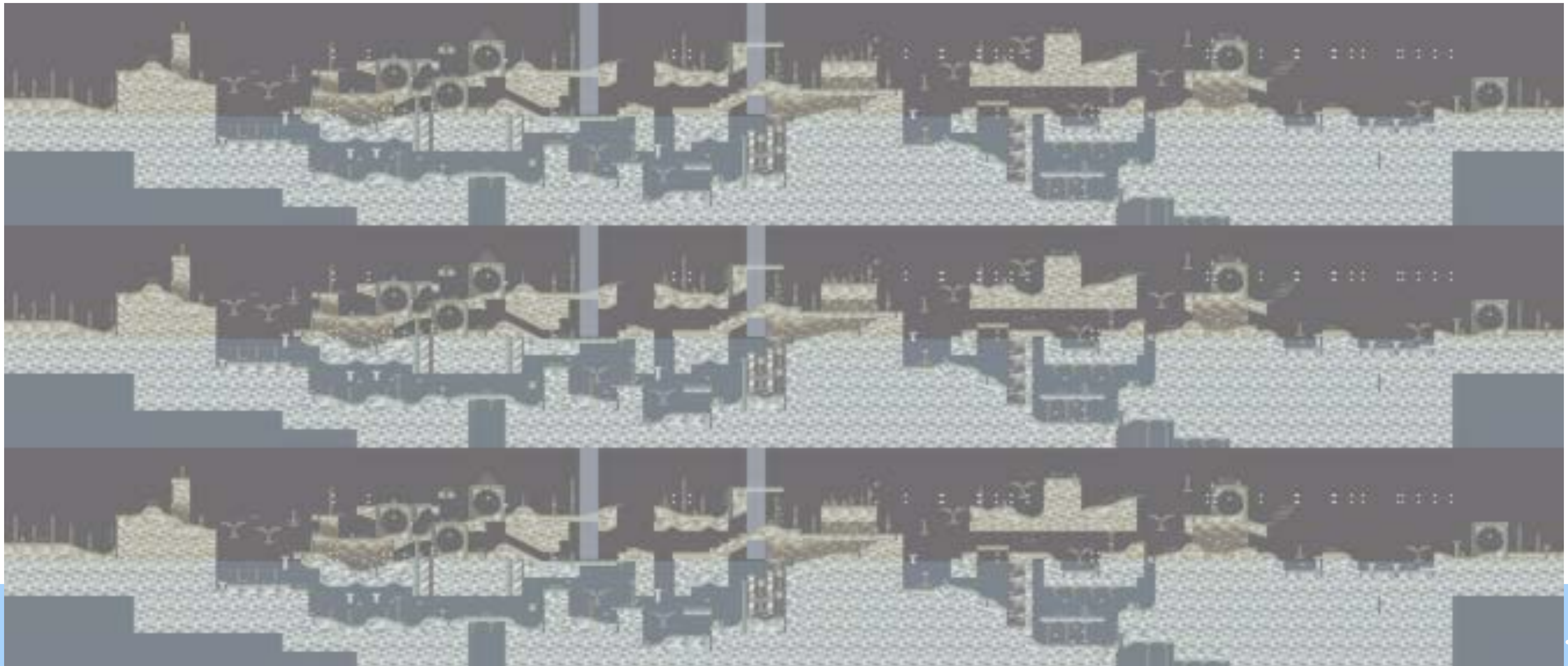
no intrinsic reward

with intrinsic reward

our agent

rank 2

rank 3



2 — Optimization

popular ways for model update

TD control

Q-learning

$$w = w + \alpha(r + \gamma \max_a Q_w(s_{t+1}, a) - Q_w(s_t, a_t)) \nabla_w Q_w(s_t, a_t)$$

policy gradient

policy gradient theorem

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

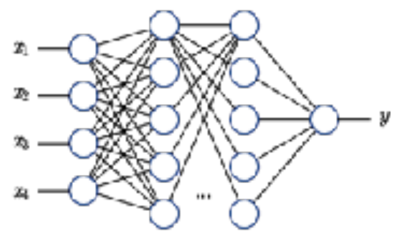
another way : derivative-free search

optimization from samples (and their evaluations)

bayesian optimization, cross-entropy,
CMA-ES, evolutionary algorithms …

(nature inspired heuristic search)

RL by derivative-free search

sample  black box → total reward evaluation

involve both model update and exploration

have shown advantages

- for decades e.g. [Shimon Whiteson. Evolutionary computation for reinforcement learning. In: Reinforcement Learning: State-of-the-Art, 2012]
- and recently e.g. [Such et al., Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. arXiv:1712.06567]

but limitations are also significant

- hard to scale up for large policy models

  e.g. [Qian et al., Derivative-free optimization of high-dimensional non-convex functions by sequential random embeddings, IJCAI'16]

- sensitive to evaluation noise

  e.g. [Wang et al., Noisy derivative-free optimization with value suppression, AAAI'18]

Gym+Mujoco environments

deal with noise noisy

Task	VS	MPS
Acrobot-v1↓	**80.76**±1.38	86.50±4.45
MountainCar-v0↓	**134.92**±3.87	150.85±13.33
HalfCheetah-v1↑	**1924.60**±278.08	1388.27±479.94
Humanoid-v1↑	**461.85**±23.92	422.40±41.84
Swimmer-v1↑	**360.51**±3.45	289.97±71.70
Ant-v1↑	**1312.85**±90.16	1126.89±123.11
Hopper-v1↑	**1111.91**±117.69	873.87±186.46
LunarLander-v2↑	**80.40**±54.51	-187.70±107.00

3 — Environment modeling

model-based RL can be much more efficient

if a good model is available

learning raw transitions is usually infeasible

the world-model [Ha & Schmidhuber, arXiv:1803.10122]
learns in the latent space of an AE by RNN
(did not work in the Retro Contest)

in between model-based and model-free

use (inaccurate) model output as state features, e.g.,

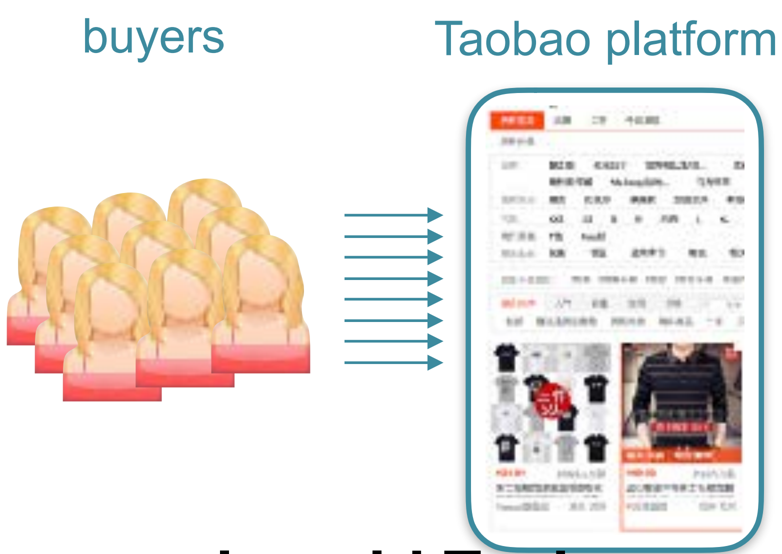
Value Iteration Network [Tamar et al., NIPS'16]
Imagination-Augmented Agents [Weber et al., arXiv:1707.06203]
…

Manually learned environment

simulators for aircraft/robot design are common

simulator for online shopping ?

- involve customers
- large uncertainty
- adaptive customer policy



real-world Taobao

customer-platform interactions

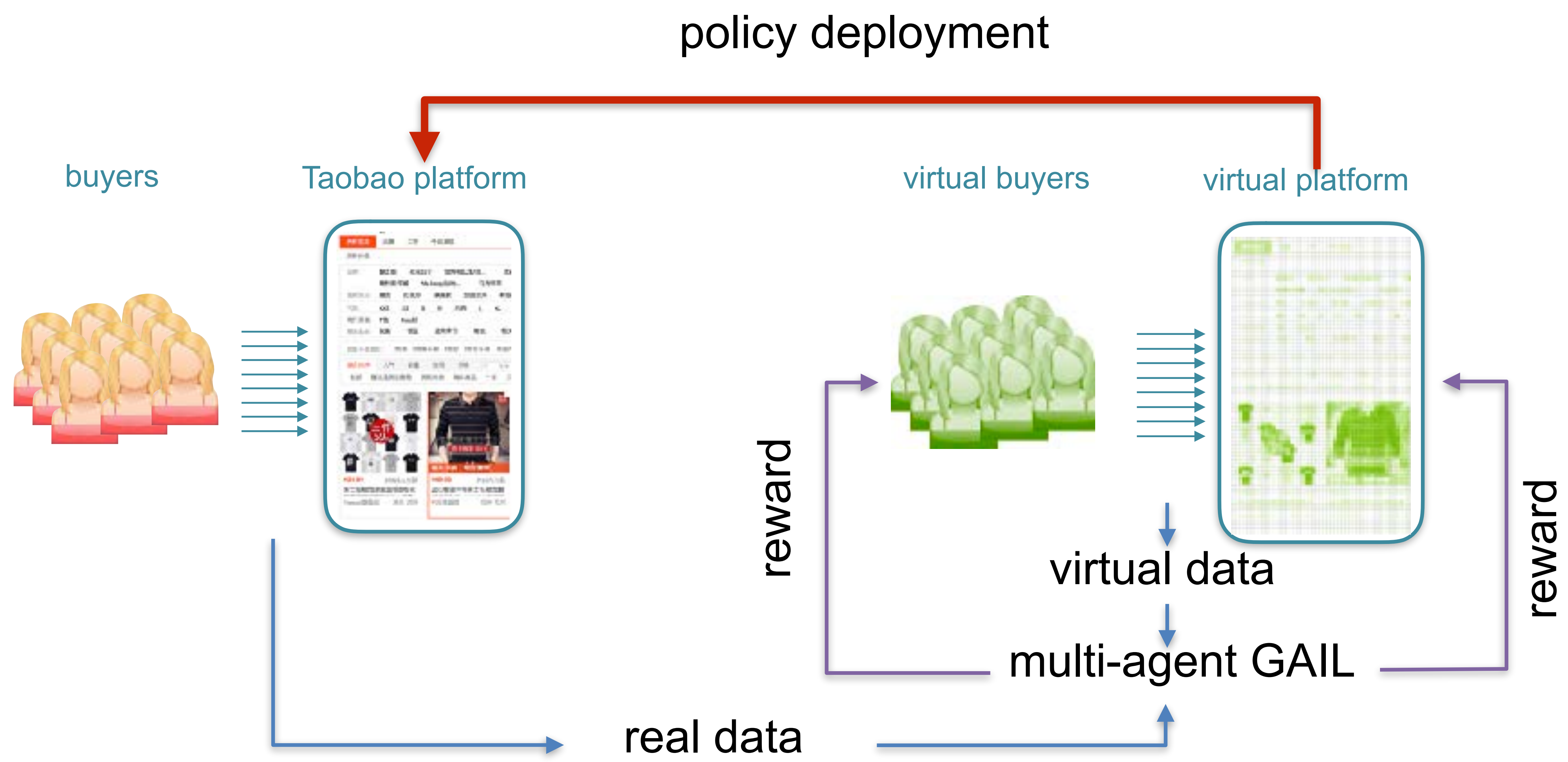
Virtual Taobao

Supervised transition learning does not work

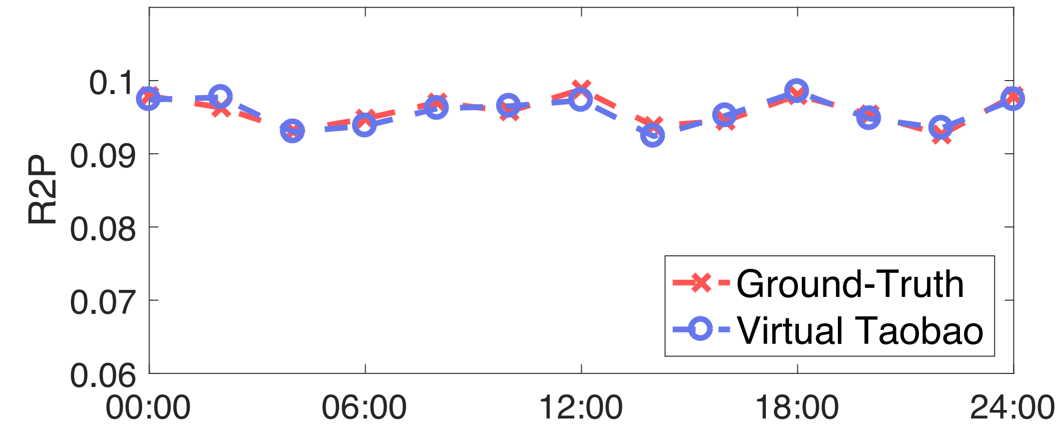
environment (customers) changes as the policy changes

Multi-agent imitation learning

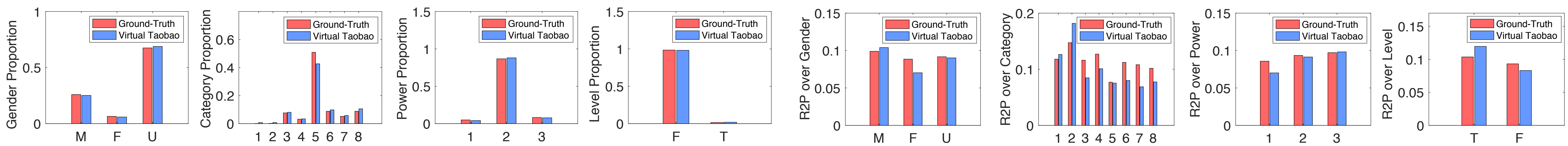
[Shi, et al. Virtual-Taobao: Virtualizing real-world online retail environment for reinforcement learning. arXiv 1805.10000]



Virtual vs. real

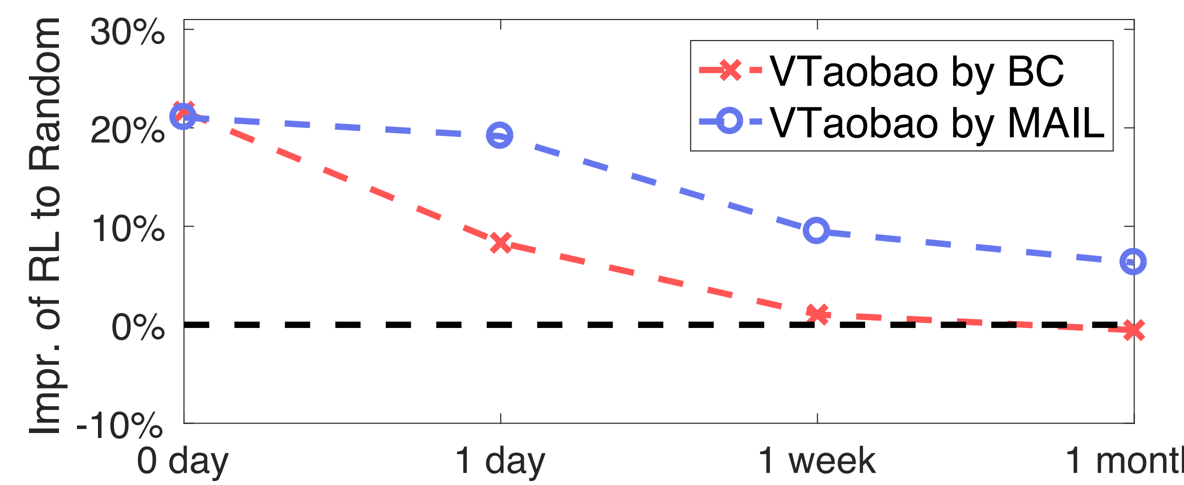


close to real environment



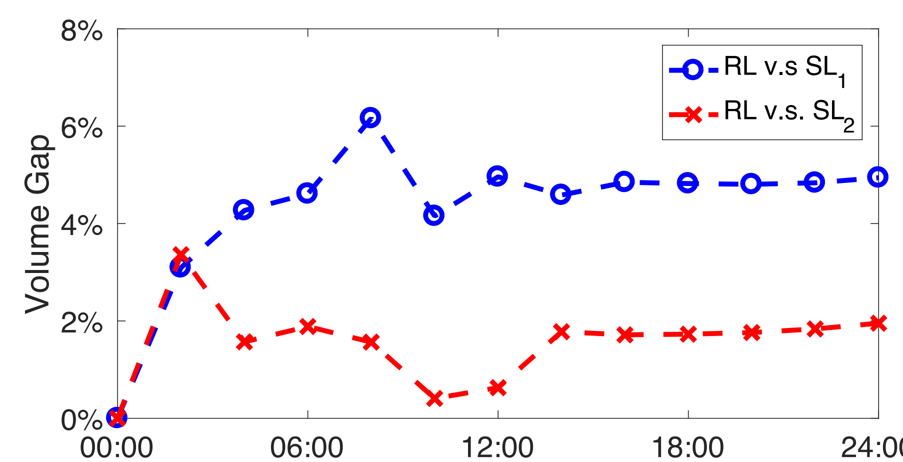
multi-agent imitation learning
vs. supervised imitation

better models environment changes



with constraints,
deliver applicable policy

2% GMV increase in A/B test



[Shi, et al. Virtual-Taobao: Virtualizing real-world online retail environment for reinforcement learning. arXiv 1805.10000]

4 — Experience transfer

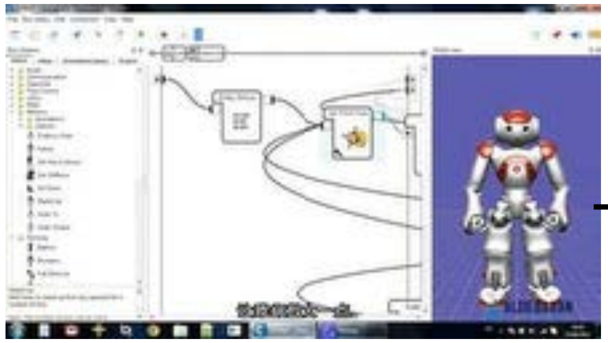
Accumulate and reuse experience is a key part of human intelligence

transfer of samples, e.g., [Lazaric et al., ICML’08]
transfer of representation, e.g., [Ferrante et al., AAMAS’08]
transfer of skills/options, e.g., [Sutton et al., AIJ’99]

transfer out of the simulators

from policy on states
to policy on state + environment features

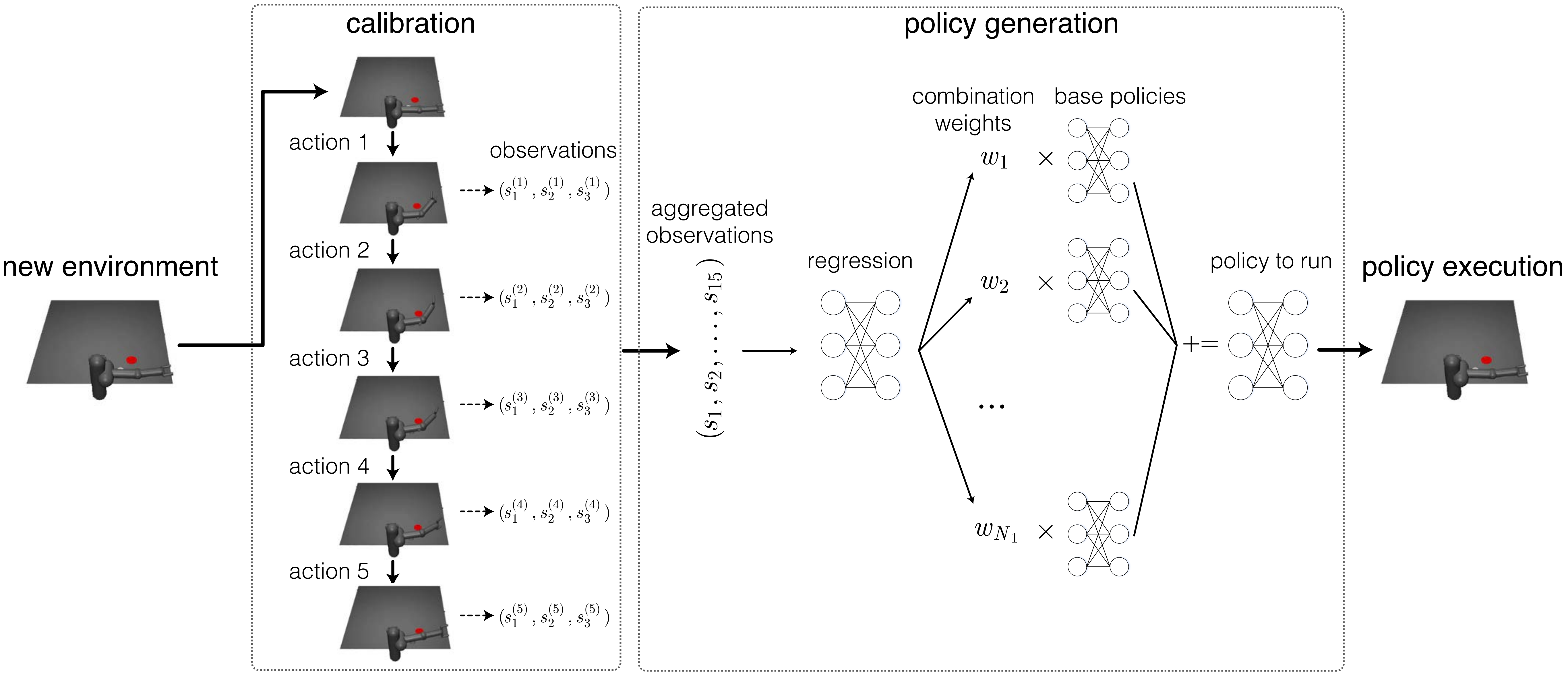
$$\pi(s) \longrightarrow \pi(s, \eta)$$



implicit feature learning [Peng et al., arXiv:1710.06537]
explicit feature learning [Zhang et al., IJCAI’18]

POSEC: Self-calibration

[Zhang, et al. Learning environmental calibration actions for policy self-evolution. IJCAI'18]
tomorrow 8:30 K2

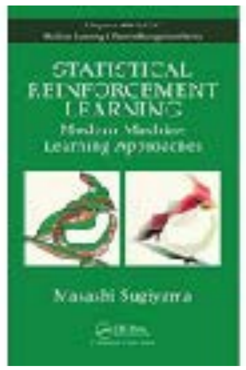


Other directions

- Partial-observable and other semi-MDP
- Hierarchical reinforcement learning
- Reward design
- …

Books

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



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

Online resources

OpenAI Gym Reinforcement Learning toolkits
https://gym.openai.com
Awesome-RL https://github.com/aikorea/awesome-rl
Resources at MST http://web.mst.edu/~gosavia/rl_website.html

Lectures by David Silver
http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html
Berkeley CS 294: Deep Reinforcement Learning
http://rll.berkeley.edu/deeprlcourse/