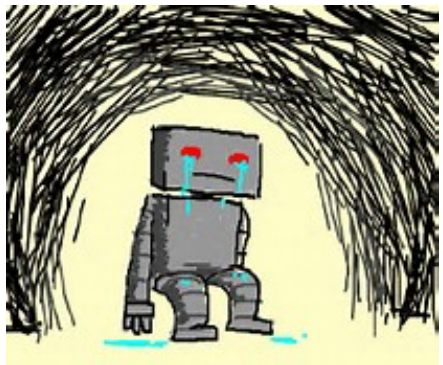
南京大学
人工智能学院
School of Artificial Intelligence, Nanjing University

Lecture 5
From MDP to Reinforcement Learning

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

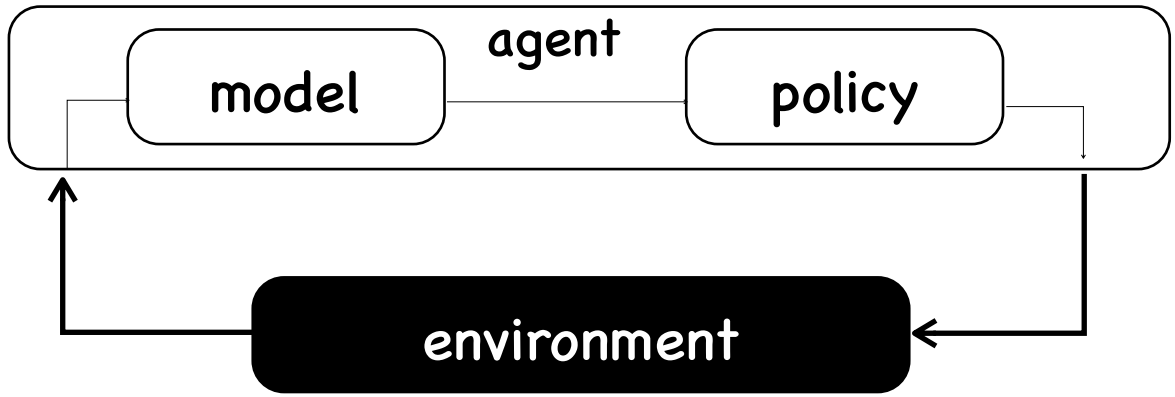
model-based

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?

...

Model-free RL

explore the environment and learn policy at the same time

Monte-Carlo method

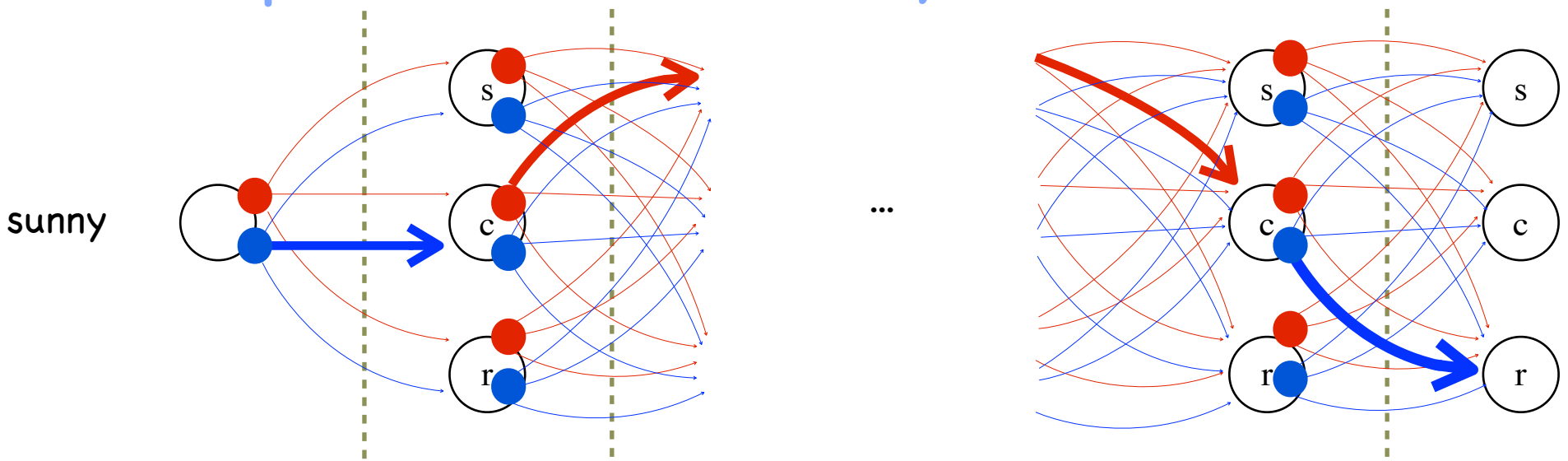
Temporal difference method

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

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

$\mathrm{c}(s_t, a_t)$++

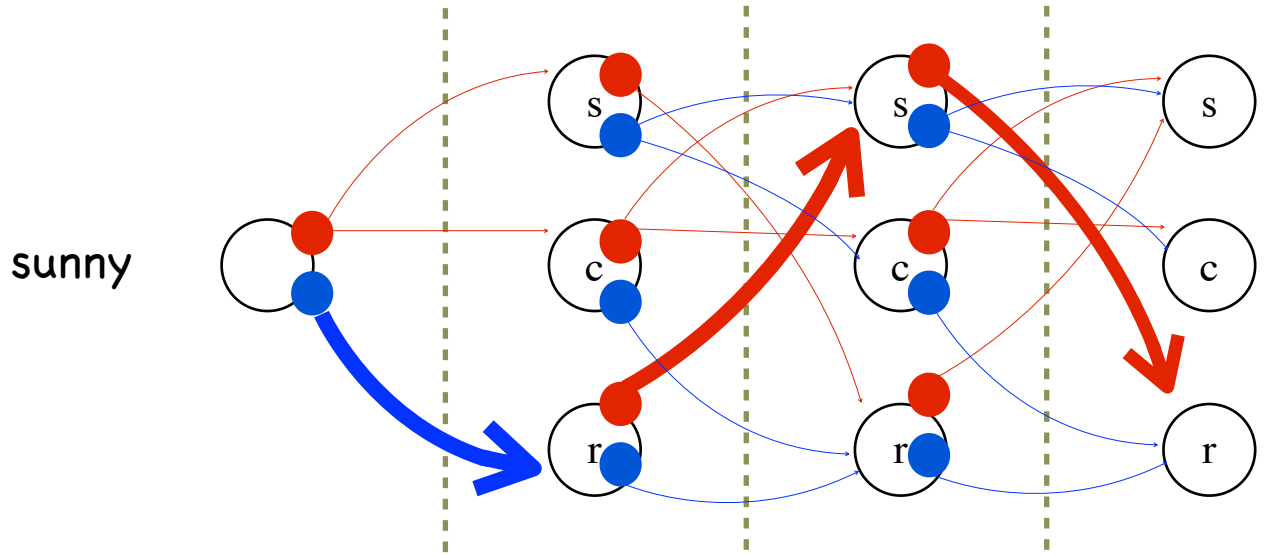
end for

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

end for

improvement ?

problem: what if the policy takes only one path?



cannot improve the policy

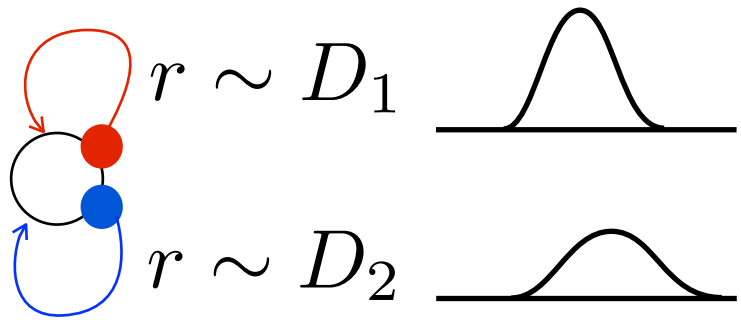
no exploration of the environment

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:

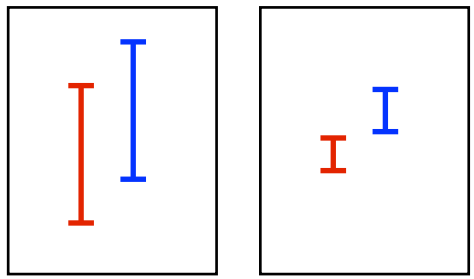
probability according to action quality

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

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

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

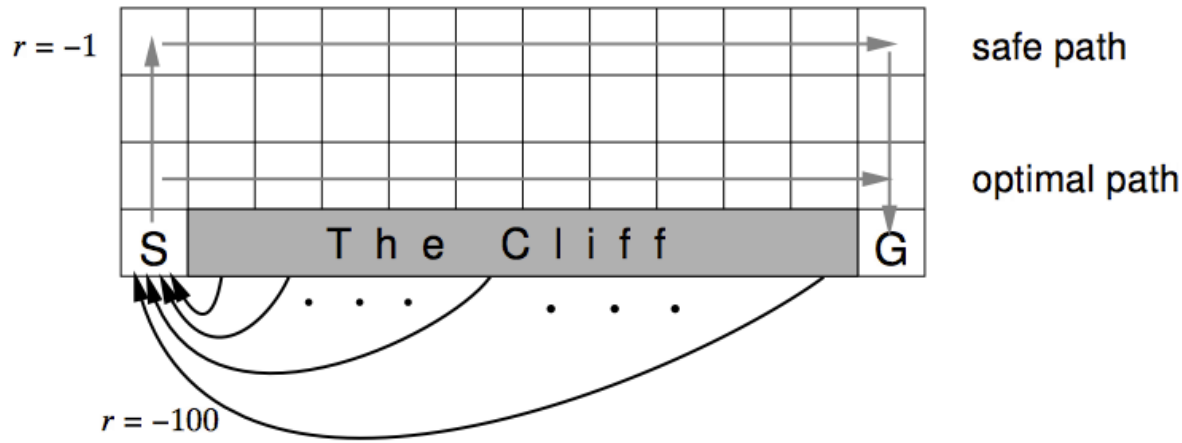
$\mathrm{c}(s_t, a_t)$++

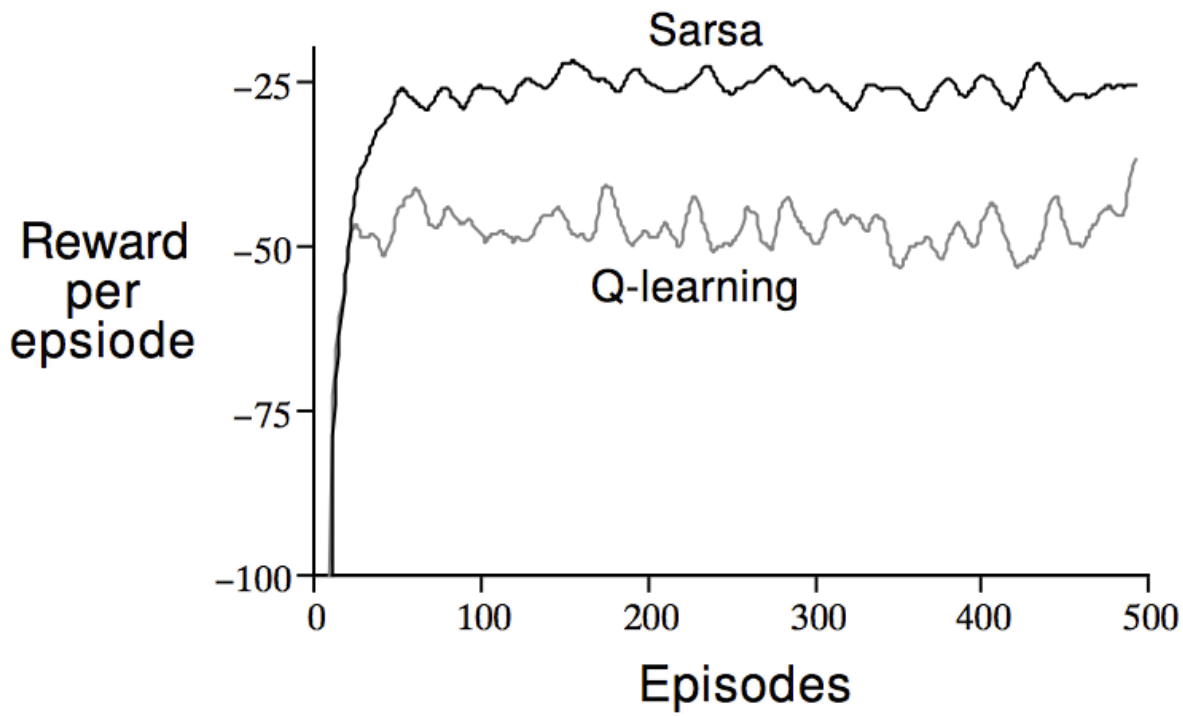
end for

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

end for

SARSA v.s. Q-learning





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

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

-- 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 $T \times \prod_{i=t+1}^{T-1} \frac{\pi(s_i, a_i)}{p_i}$

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

$\mathrm{c}(s_t, a_t)++$

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:

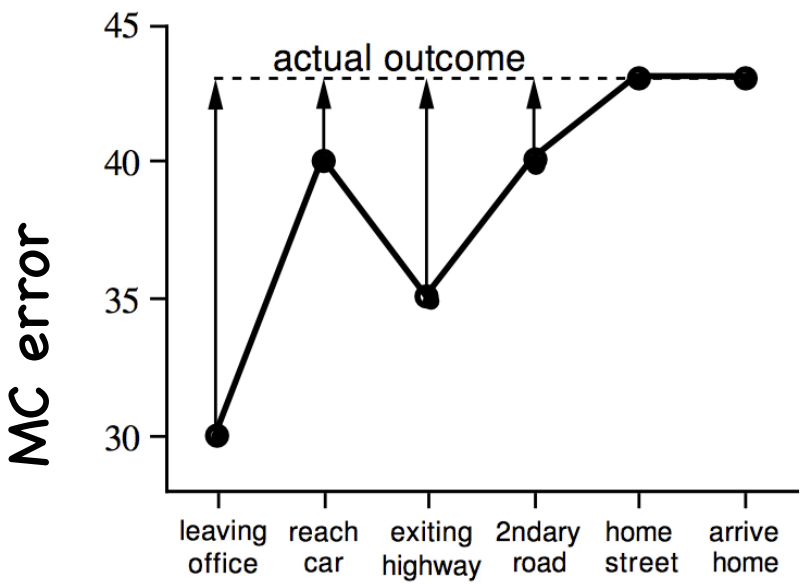
$$Q(s_t, a_t) \Leftarrow Q(s_t, a_t) + \alpha(\underbrace{R - Q(s_t, a_t)}_{\text{MC error}})$$

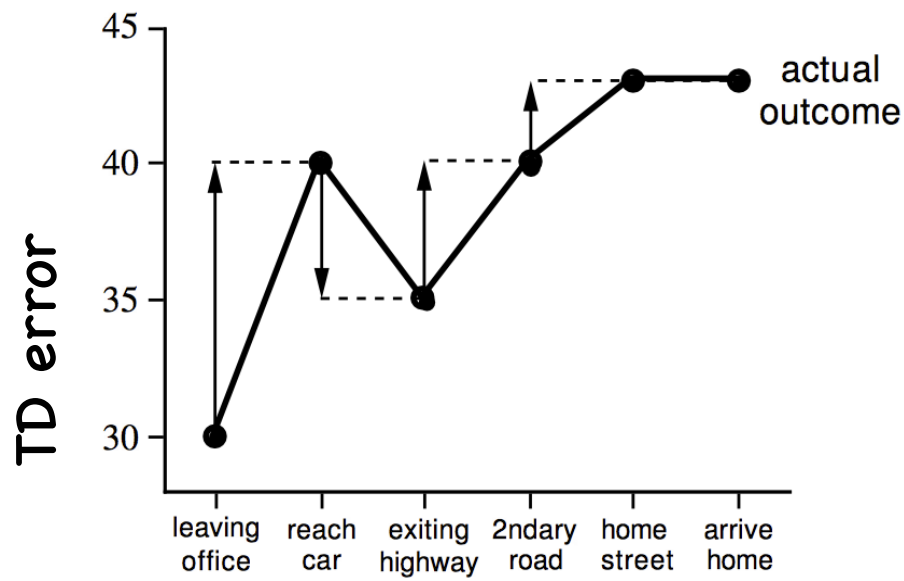
TD update:

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

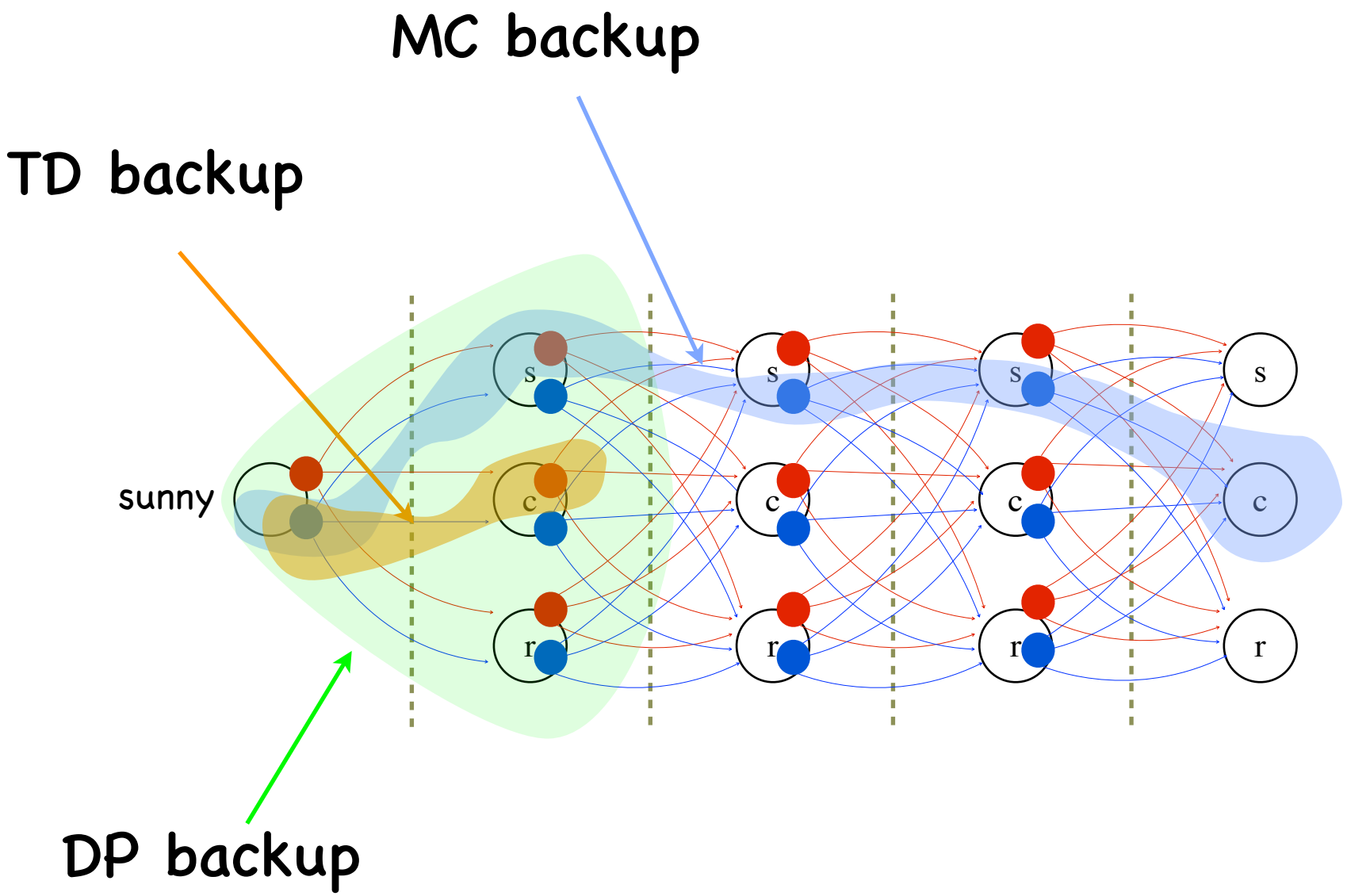
Temporal-Difference Learning - example

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



On-policy TD control

$Q_0 = 0$, initial state

for i=0, 1, ...

$a = \pi_\epsilon(s)$

s', r = do action a

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

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

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

$s = s$'

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' = $\pi(s')$

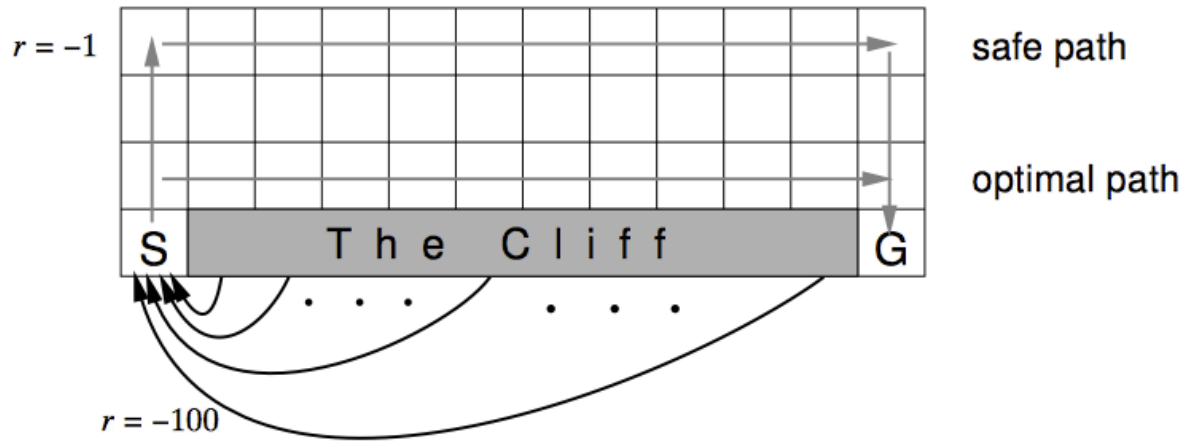
$Q(s,a)$+= $\alpha(r + \gamma Q(s',a') - Q(s,a))$

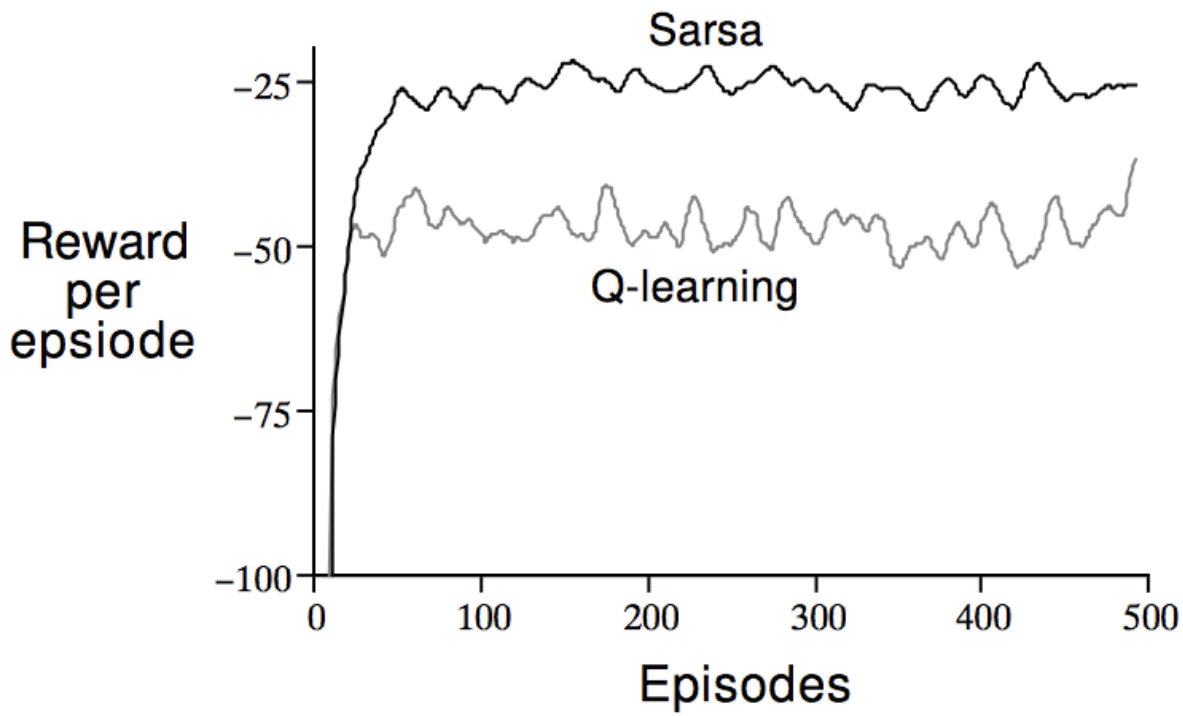
$\pi(s) = \arg\max_a Q(s,a)$

$s = s$'

end for

SARSA v.s. Q-learning

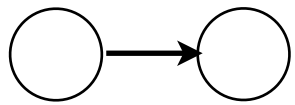




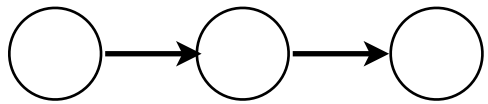
λ-return

in between TD and MC: n-step prediction

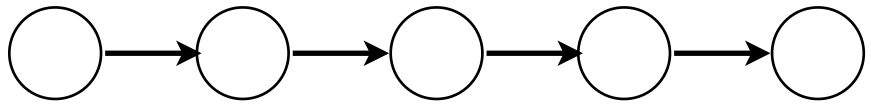
n-step return

TD(1-step) 

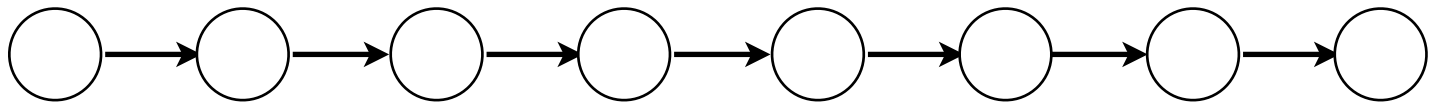
$$R^{(1)} = r_{t+1} + \gamma Q(s_{t+1}, a_{t+1})$$

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 

k-step TD:

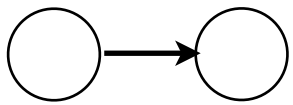
$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha(R^{(k)} - Q(s_t, a_t))$$

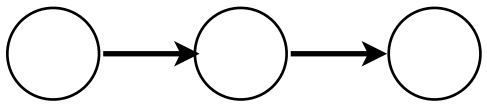
$$R^{(\max)} = \sum_{i=1}^{T} \gamma^{i-1} r_{t+i}$$

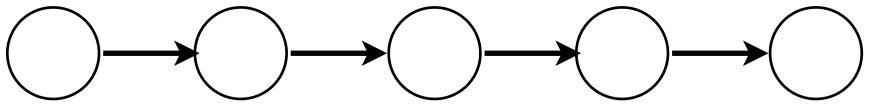
λ-return

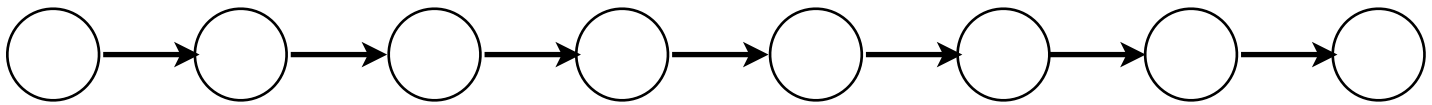
averaging k-step returns, parameter λ

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

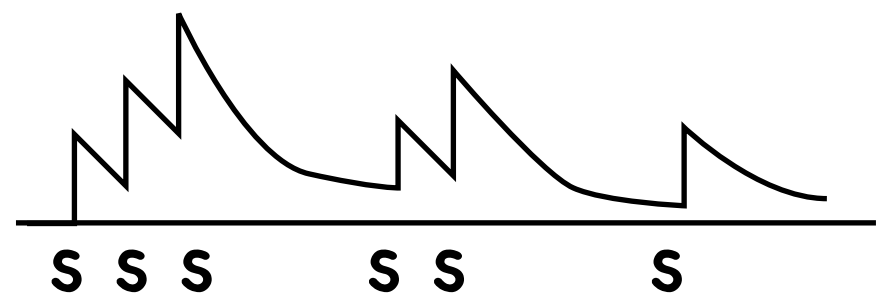
Implementation: eligibility traces

Maintain an extra memory E(s)

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

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

$E(s)$



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