Reinforcement learning

Markov decision problem

MDP consists of:

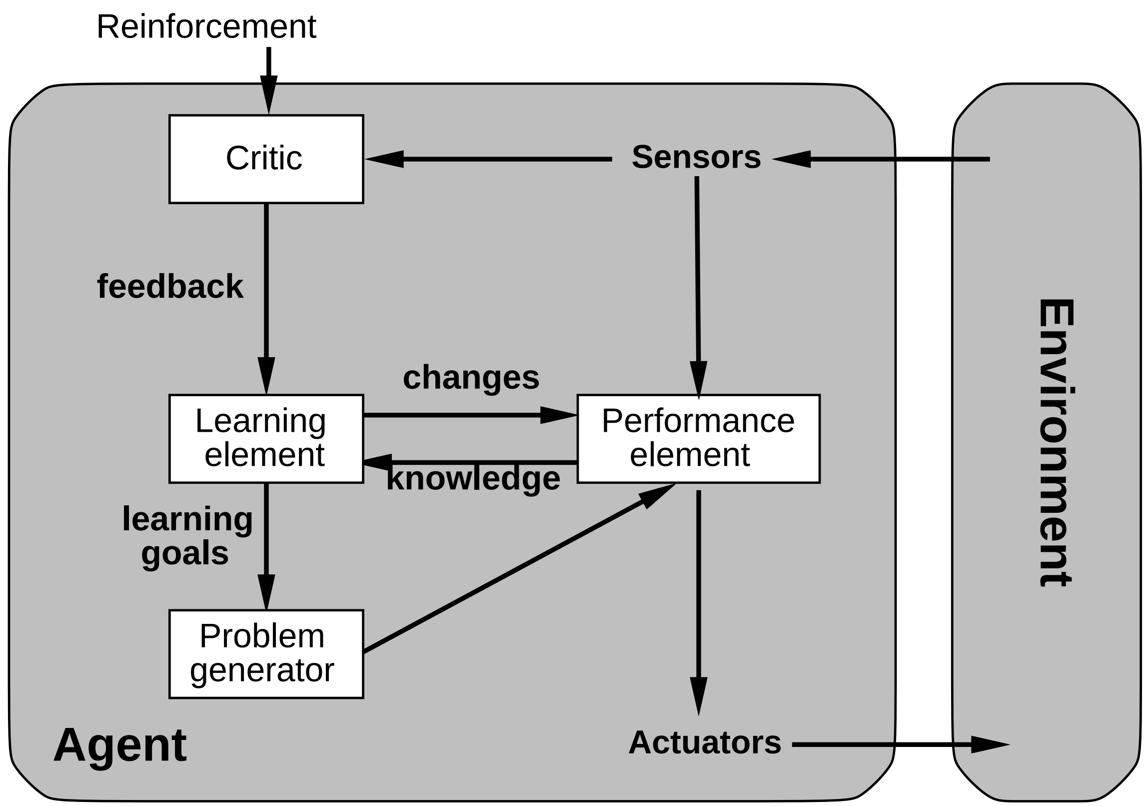
States $s \in S$
Actions $a \in A$
Model $T(s, a, s') \equiv P(s'|s, a)$ = probability that a in s leads to s'
Reward function $R(s)$

Previous topic: determine optimal policy, $\pi^*(s)$ given $T(s, a, s')$ and $R(s)$

Next topic: determine optimal policy, $\pi^*(s)$ when $T(s, a, s')$ and $R(s)$ are unknown.

E.g., navigating an unfamiliar city
E.g., pole balancing

Interaction with an unknown environment



The reinforcement learning problem

At each time step t, the agent is in some state s_t.

Agent must choose an action a_t.

Action causes state update $s_{t+1} = \delta(s_t, a_t)$
and agent receives reward $r(s_{t+1})$

Passive reinforcement learning

Agent's policy π is fixed; goal is to learn utility function

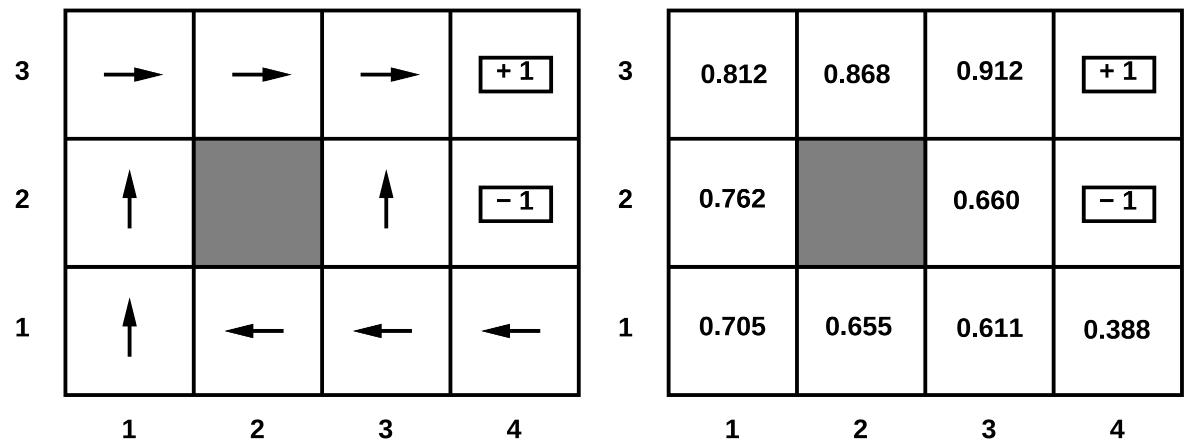
$$U^\pi(s) = E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t) | \pi, s_0 = s\right]$$

Cannot use value iteration algorithm

$$U(s) \leftarrow R(s) + \gamma \max_a \Sigma_{s'} U(s') T(s, a, s') \qquad \text{for all } s$$

because T and R are unknown.

Passive reinforcement learning (contd.)



Idea: Run a series of trials

$(1,1)_{-.04} \rightarrow (1,2)_{-.04} \rightarrow (1,3)_{-.04} \rightarrow (1,2)_{-.04} \rightarrow (1,3)_{-.04} \rightarrow (2,3)_{-.04} \rightarrow (3,3)_{-.04} \rightarrow (4,3)_{+1}$

$(1,1)_{-.04} \rightarrow (1,2)_{-.04} \rightarrow (1,3)_{-.04} \rightarrow (2,3)_{-.04} \rightarrow (3,3)_{-.04} \rightarrow (3,2)_{-.04} \rightarrow (3,3)_{-.04} \rightarrow (4,3)_{+1}$

$(1,1)_{-.04} \rightarrow (2,1)_{-.04} \rightarrow (1,1)_{-.04} \rightarrow (1,2)_{-.04} \rightarrow (1,3)_{-.04} \rightarrow (2,3)_{-.04} \rightarrow (3,3)_{-.04} \rightarrow (3,2)_{-.04} \rightarrow (3,4)_{-1}$

Scheme 1: Direct estimation of utility

Compute expectation over observed state sequences:

$$U^\pi(s) = E_{sample}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t) | \pi, s_0 = s\right]$$

Problem: Fails to exploit knowledge about how states are connected.

Utility of a state s is related to expected utility of successor states s':

$$U^\pi(s) = R(s) + \gamma \sum_{s'} T(s, \pi(s), s') U^\pi(s')$$

Previously, much experience with s_1:

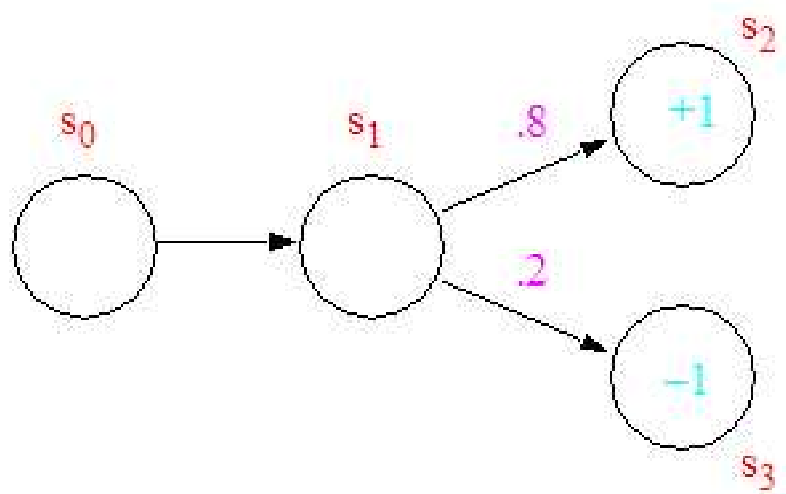
$s_7 \rightarrow s_1 \rightarrow s_2$

$s_4 \rightarrow s_1 \rightarrow s_3$

$s_4 \rightarrow s_1 \rightarrow s_2$

$s_5 \rightarrow s_1 \rightarrow s_2$

Now, $s_0 \rightarrow s_1 \rightarrow s_2$.



Scheme 2: Adaptive dynamic programming

Learn $T(s, \pi(s), s')$ and $R(s)$ and then apply ordinary value iteration.

How do we learn?

Keep track of $N(s, \pi(s), s')$, the count of the number of times the policy took agent from s to s'.

$$\hat{T}(s, \pi(s), s') = N(s, \pi(s), s') / \sum_{x} N(s, \pi(s), x)$$

Keep track of $r(s)$, the total reinforcement received in state s.

$$\hat{R}(s) = r(s) / \sum_{x,y} N(s, y, x)$$

Model based versus direct (model free)

Scheme 3: Temporal difference (TD) learning

Direct method that exploits the identity

$$U^\pi(s) = R(s) + \gamma \sum_{s'} T(s, \pi(s), s') U^\pi(s')$$

or, if state transitions are deterministic,

$$U^\pi(s) = R(s) + \gamma U^\pi(s')$$

Use observed transitions to adjust values of observed states to agree with the identity:

$$U^\pi(s) \leftarrow R(s) + \gamma U^\pi(s')$$

Because rewards and transitions can be nondeterministic, don't simply replace utility estimate, average old and new:

$$U^\pi(s) \leftarrow (1 - \xi) U^\pi(s) + \xi [R(s) + \gamma U^\pi(s')] \text{ with } \xi \in [0, 1]$$

$$U^\pi(s) \leftarrow U^\pi(s) + \xi [R(s) + \gamma U^\pi(s') - U^\pi(s)]$$

Active reinforcement learning

Active ≡ choice of action is not given; must be learned.

I.e., find π^* that maximizes cumulative reward.

Active greedy reinforcement learning

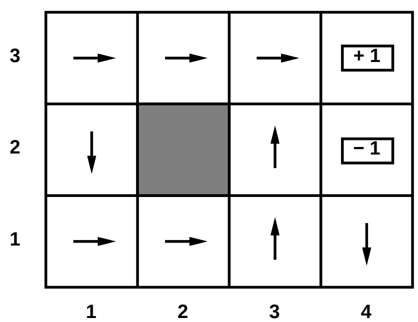
— start with random policy

— use ADP to estimate world model and utility function

— use utility function and one-step lookahead to update policy

— repeat

Initial policy has a big impact on ultimate policy →
agent seldom discovers optimal policy



Exploration - Exploitation Dilemma

Should we use current policy, or try out alternative actions to see if they are better?

E.g., exploring a new city

Possibilities:

— with probability μ, choose random action instead of action prescribed under current policy

— softmax: $P(a|s) = \alpha \Sigma_{s'} T(s, a, s') \exp(U(s')/\nu)$

— initially μ or ν large, but decrease over time (stationary env.)

— initialize utility function with optimistic estimates of utility (any unexplored state will be preferred over an explored state)

E.g., eating

Lousy strategy will reflect utilities under current policy

Q values

Active ADP agent constructs explicit model of environment— $T(s, a, s')$ and $R(s)$.

Direct alternative to model-based approach: Q values

$Q(a, s)$: Utility of taking action a in state s
$U(s) = \max_a Q(a, s)$

$Q(a, s) = R(s) + \gamma \sum_{s'} T(s, a, s') U(s')$ for all s, a
— immediate reward received upon executing action a in state s,
plus discounted utility of following optimal policy thereafter.

Equivalently,
$Q(a, s) = R(s) + \gamma \sum_{s'} T(s, a, s') \max_{a'} Q(a', s')$

Q learning

Given

$$Q(a,s) = R(s) + \gamma \textstyle\sum_{s'} T(s,a,s') \max_{a'} Q(s',a')$$

use TD updating procedure on Q given observed sequence of states and rewards:

$$Q(a,s) \leftarrow (1-\xi)Q(a,s) + \xi[R(s) + \gamma \max_{a'} Q(a',s')]$$

Requires exploration strategy!

Optimal policy:

$$\pi^*(s) = \mathsf{argmax}_a Q(a,s)$$

Comments on Q learning

Theorem (Watkins & Dayan, 1992): Q-learning will eventually converge to the optimal policy for any deterministic MDP

Theorem (Sutton, 1988; Dayan, 1992): TD-learning will also converge with probability 1.

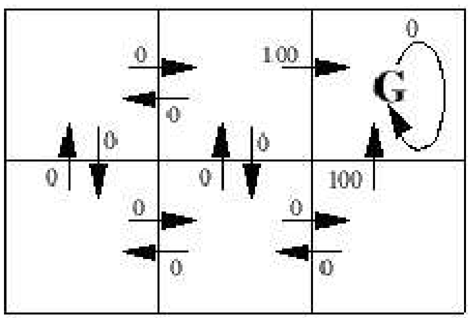
Convergence is slow if search space is large:
Theorem relies on visiting every state infinitely often

For real-world problems, can't rely on a look up table for $Q(a, s)$;
need to have some type of generalization across states

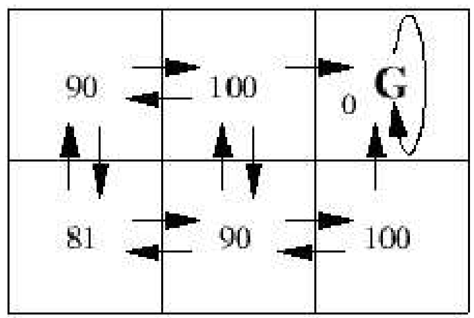
Q learning example

From T.Mitchell, *Machine Learning*, McGraw-Hill 1997.

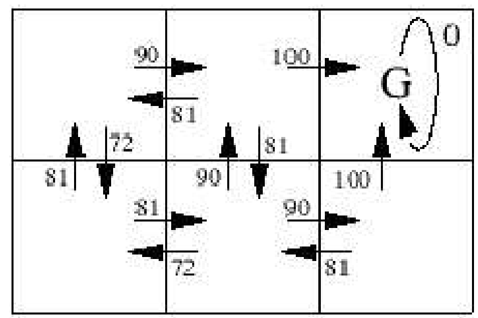
Assume $\gamma = 0.9$



Reward



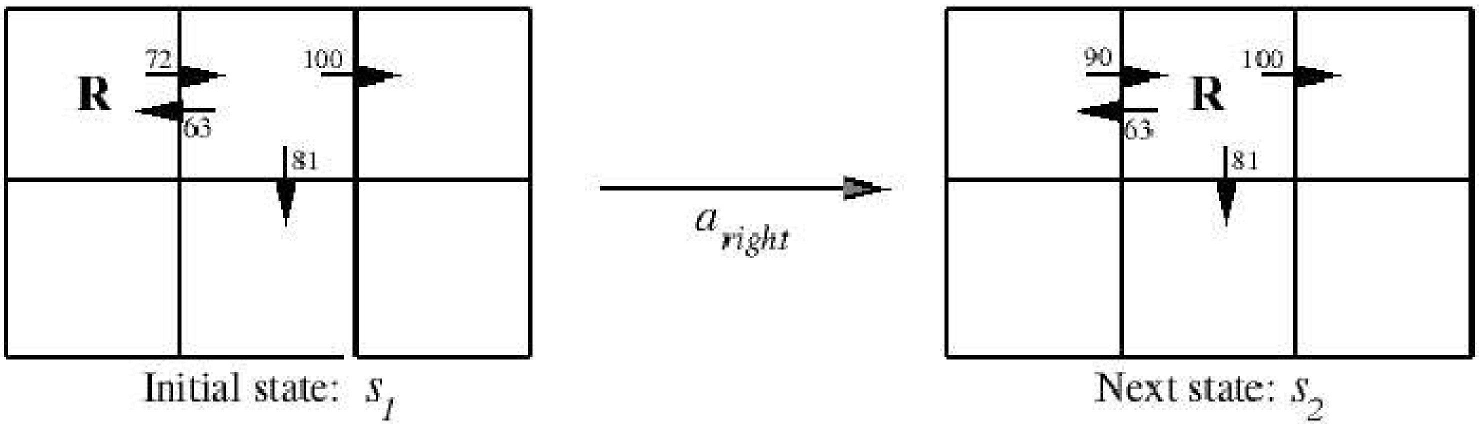
Value



$Q(s, a)$ values

Q learning example

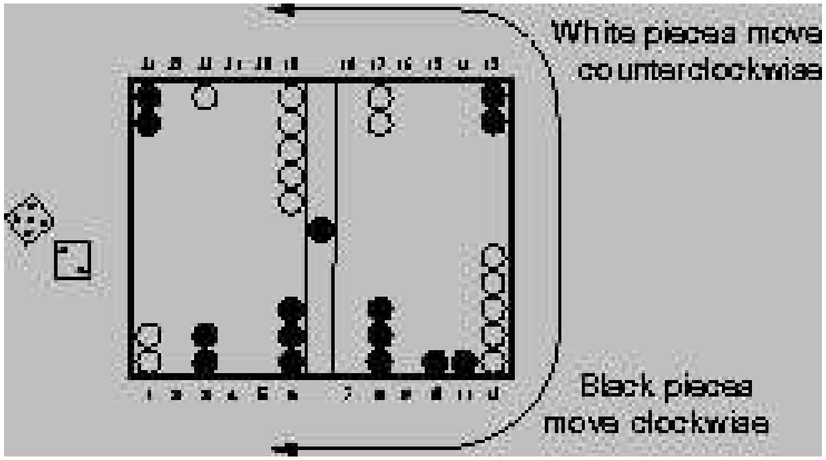
From T.Mitchell, *Machine Learning*, McGraw-Hill 1997.



$$
\begin{aligned}
\hat{Q}(s_1, a_{right}) &\leftarrow r + \gamma \max_{b} \hat{Q}(s_2, b) \\
&\leftarrow 0 + 0.9 \max\{63, 81, 100\} \\
&\leftarrow 90
\end{aligned}
$$

TD Gammon

Learns to play backgammon with temporal-difference estimation



Program	Hidden Units	Training Games	Opponents	Results
TD-Gam 0.0	40	300,000	Other Programs	Tied for Best
TD-Gam 1.0	80	300,000	Robertie, Magriel, ...	−13 pts / 51 games
TD-Gam 2.0	40	800,000	Var. Grandmasters	−7 pts / 38 games
TD-Gam 2.1	80	1,500,000	Robertie	−1 pts / 40 games
TD-Gam 3.0	80	1,500,000	Kazaros	+6 pts / 20 games

TD Gammon

Active reinforcement learning, in which transition and reward models known.

A variation on value iteration:
— $U(s)$ updated via TD procedure
— $U(s)$ approximated with neural net instead of look up table
— policy optimized by choosing action that maximizes utility
— exploration strategy
— TD(λ) with $\lambda = .7$,

λ: how much look ahead in estimating utility

$$
\begin{array}{ll}
U^\pi(s) = R(s) + \gamma U^\pi(s') & \lambda = 0 \\
U^\pi(s) = R(s) + \gamma R(s') + \gamma^2 U^\pi(s'') & . \\
U^\pi(s) = R(s) + \gamma R(s') + \gamma^2 R(s'') + \gamma^3 U^\pi(s''') & . \\
\ldots & . \\
U^\pi(s) = \Sigma_t \gamma^t R(s_t) & \lambda = 1
\end{array}
$$

Issues

— active or passive reinforcement learning
— explicit or implicit model of environment