CSCE 478/878 Lecture 10: Reinforcement Learning

Stephen D. Scott
(Adapted from Tom Mitchell's slides)

December 1, 2004

Outline

- Control learning
- Control policies that choose optimal actions
- Q learning
- Convergence
- Temporal difference learning

Control Learning

Consider learning to choose actions, e.g.

- Robot learning to dock on battery charger
- Learning to choose actions to optimize factory output
- Learning to play Backgammon

Note several problem characteristics:

- Delayed reward (thus have problem of temporal credit assignment)
- Opportunity for active exploration (versus exploitation of known good actions)
- Possibility that state only partially observable

Example: TD-Gammon

[Tesauro, 1995]

Learn to play Backgammon

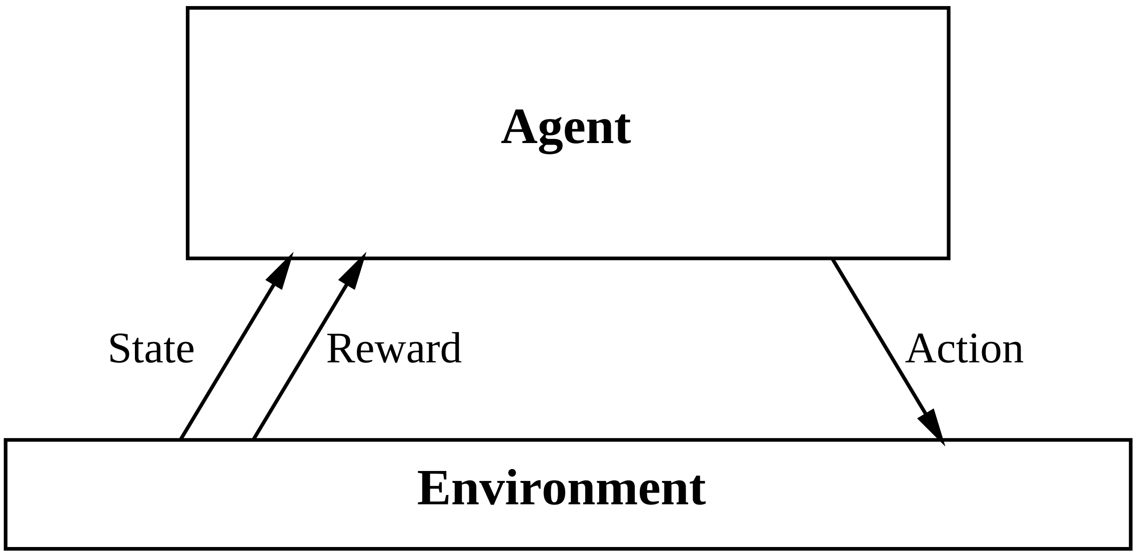
Immediate Reward:

- $+100$ if win
- -100 if lose
- 0 for all other states

Trained by playing 1.5 million games against itself

Now approximately equal to best human player

Reinforcement Learning Problem



$$s_0 \xrightarrow[r_0]{a_0} s_1 \xrightarrow[r_1]{a_1} s_2 \xrightarrow[r_2]{a_2} \ldots$$

Goal: Learn to choose actions that maximize

$$r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots \text{ , where } 0 \leq \gamma < 1$$

Markov Decision Processes

Assume

- Finite set of states S
- Set of actions A
- At each discrete time agent observes state $s_t \in S$ and chooses action $a_t \in A$
- Then receives immediate reward r_t, and state changes to s_{t+1}
- Markov assumption: $s_{t+1} = \delta(s_t, a_t)$ and $r_t = r(s_t, a_t)$
  - I.e. r_t and s_{t+1} depend only on <u>current</u> state and action
  - Functions δ and r may be nondeterministic
  - Functions δ and r not necessarily known to agent

Agent's Learning Task

Execute actions in environment, observe results, and

- learn action policy $\pi : S \rightarrow A$ that maximizes

$$E\left[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots\right]$$

  from any starting state in S

- Here $0 \leq \gamma < 1$ is the discount factor for future rewards

Note something new:

- Target function is $\pi : S \rightarrow A$
- But we have no training examples of form $\langle s, a \rangle$
- Training examples are of form $\langle \langle s, a \rangle, r \rangle$
- I.e. not told what best action is (e.g. checkers in Chapt. 1), instead told reward for executing action a in state s

Value Function

First consider deterministic worlds

For each possible policy π the agent might adopt, we can define an evaluation function over states

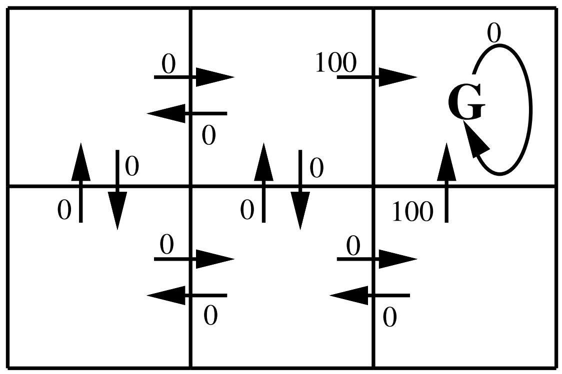
$$\begin{aligned} V^{\pi}(s) &\equiv r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots \\ &\equiv \sum_{i=0}^{\infty} \gamma^i r_{t+i} \end{aligned}$$

where $r_t, r_{t+1}, \ldots$ are generated by following policy π, starting at state s

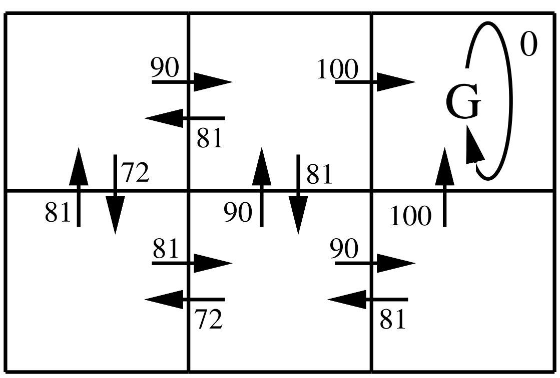
Restated, the task is to learn the optimal policy π^*

$$\pi^* \equiv \underset{\pi}{\text{argmax}}\ V^{\pi}(s), \quad (\forall s)$$

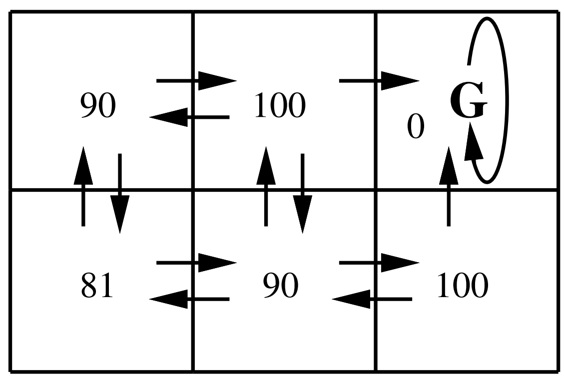
Value Function
(cont'd)



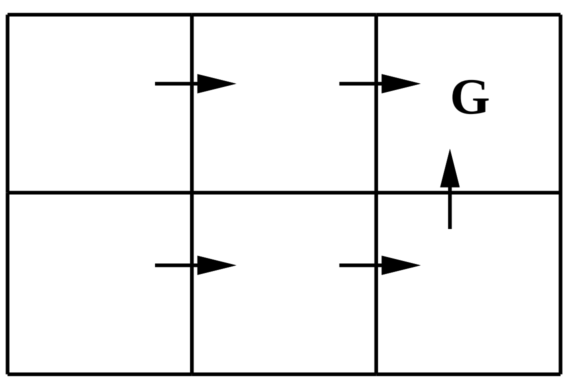
$r(s, a)$ (immediate reward) values



$Q(s, a)$ values



$V^*(s)$ values



One optimal policy

What to Learn

We might try to have agent learn the evaluation function V^{π^*} (which we write as V^*), i.e. what checkers player tried

It could then do a lookahead search to choose best action from any state s because

$$\pi^*(s) = \operatorname*{argmax}_a \left[r(s,a) + \gamma V^*(\delta(s,a))\right]$$

i.e. choose action that maximized immediate reward $+$ discounted reward if optimal strategy followed from then on

E.g. $V^*(bot.\ ctr.) = 0 + \gamma 100 + \gamma^2 0 + \gamma^3 0 + \cdots = 90$

A problem:

- This works well if agent knows $\delta : S \times A \rightarrow S$, and $r : S \times A \rightarrow \Re$
- But when it doesn't, it can't choose actions this way

Q Function

Define new function very similar to V^*:

$$Q(s,a) \equiv r(s,a) + \gamma V^*(\delta(s,a))$$

i.e. $Q(s,a) =$ total discounted reward if action a taken in state s and optimal choices made from then on

If agent learns Q, it can choose optimal action even without knowing δ!

$$\begin{aligned} \pi^*(s) &= \operatorname*{argmax}_a \left[r(s,a) + \gamma V^*(\delta(s,a))\right] \\ &= \operatorname*{argmax}_a Q(s,a) \end{aligned}$$

Q is the evaluation function the agent will learn

Training Rule to Learn Q

Note Q and V^* closely related:

$$V^*(s) = \max_{a'} Q(s, a')$$

Which allows us to write Q recursively as

$$\begin{aligned} Q(s_t, a_t) &= r(s_t, a_t) + \gamma V^*(\delta(s_t, a_t))) \\ &= r(s_t, a_t) + \gamma \max_{a'} Q(s_{t+1}, a') \end{aligned}$$

Nice! Let $\hat{Q}$ denote learner's current approximation to Q. Consider training rule

$$\hat{Q}(s, a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s', a')$$

where s' is the state resulting from applying action a in state s

Q Learning for Deterministic Worlds

For each s, a initialize table entry $\hat{Q}(s, a) \leftarrow 0$

Observe current state s

Do forever:

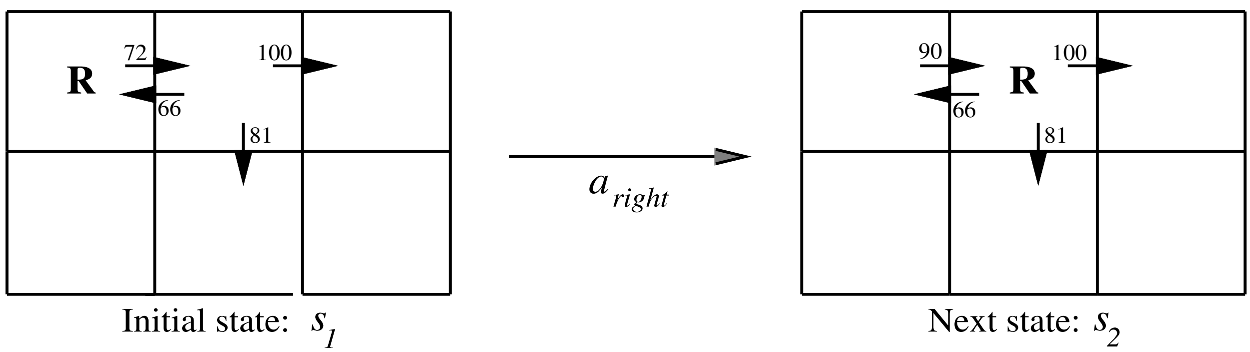
- Select an action a (greedily or probabilistically) and execute it
- Receive immediate reward r
- Observe the new state s'
- Update the table entry for $\hat{Q}(s, a)$ as follows:

$$\hat{Q}(s, a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s', a')$$

- $s \leftarrow s'$

Note that actions not taken and states not seen don't get explicit updates (might need to generalize)

Updating $\hat{Q}$



$$\begin{aligned} \hat{Q}(s_1, a_{right}) &\leftarrow r + \gamma \max_{a'} \hat{Q}(s_2, a') \\ &= 0 + 0.9 \max\{66, 81, 100\} \\ &= 90 \end{aligned}$$

Notice if rewards non-negative and $\hat{Q}$'s initially 0, then

$$(\forall s, a, n) \;\; \hat{Q}_{n+1}(s, a) \geq \hat{Q}_n(s, a)$$

and

$$(\forall s, a, n) \;\; 0 \leq \hat{Q}_n(s, a) \leq Q(s, a)$$

(can show via induction on n, using slides 11 and 12)

Updating $\hat{Q}$

Convergence

$\hat{Q}$ converges to Q. Consider case of deterministic world where each $\langle s, a \rangle$ is visited infinitely often.

Proof: Define a full interval to be an interval during which each $\langle s, a \rangle$ is visited. Will show that during each full interval the largest error in $\hat{Q}$ table is reduced by factor of γ

Let $\hat{Q}_n$ be table after n updates, and Δ_n be the maximum error in $\hat{Q}_n$; i.e.

$$\Delta_n = \max_{s,a} |\hat{Q}_n(s, a) - Q(s, a)|$$

Let $s' = \delta(s, a)$

Updating $\hat{Q}$

Convergence (cont'd)

For any table entry $\hat{Q}_n(s,a)$ updated on iteration $n+1$, error in the revised estimate $\hat{Q}_{n+1}(s,a)$ is

$$
\begin{aligned}
|\hat{Q}_{n+1}(s,a) - Q(s,a)| &= |(r + \gamma \max_{a'} \hat{Q}_n(s',a')) \\
&\quad -(r + \gamma \max_{a'} Q(s',a'))| \\
&= \gamma |\max_{a'} \hat{Q}_n(s',a') - \max_{a'} Q(s',a')| \\
(*) \quad &\leq \gamma \max_{a'} |\hat{Q}_n(s',a') - Q(s',a')| \\
(**) \quad &\leq \gamma \max_{s'',a'} |\hat{Q}_n(s'',a') - Q(s'',a')| \\
&= \gamma \Delta_n
\end{aligned}
$$

$(*)$ works since $|\max_a f_1(a) - \max_a f_2(a)| \leq \max_a |f_1(a) - f_2(a)|$

$(**)$ works since max will not decrease

Also, $\hat{Q}_0(s,a)$ bounded and $Q(s,a)$ bounded $\forall\, s,a \Rightarrow \Delta_0$ bounded

Thus after k full intervals, error $\leq \gamma^k \Delta_0$

Finally, each $\langle s,a \rangle$ visited infinitely often $\Rightarrow$ number of intervals infinite, so $\Delta_n \rightarrow 0$ as $n \rightarrow \infty$

Nondeterministic Case

What if reward and next state are non-deterministic?

We redefine V, Q by taking expected values:

$$\begin{aligned} V^{\pi}(s) &\equiv E\left[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots\right] \\ &= E\left[\sum_{i=0}^{\infty} \gamma^i r_{t+i}\right] \end{aligned}$$

$$\begin{aligned} Q(s,a) &\equiv E\left[r(s,a) + \gamma V^*(\delta(s,a))\right] \\ &= E\left[r(s,a)\right] + \gamma E\left[V^*(\delta(s,a))\right] \\ &= E\left[r(s,a)\right] + \gamma \sum_{s'} P(s' \mid s,a)\, V^*(s') \\ &= E\left[r(s,a)\right] + \gamma \sum_{s'} P(s' \mid s,a) \max_{a'} Q(s',a') \end{aligned}$$

Nondeterministic Case
(cont'd)

Q learning generalizes to nondeterministic worlds

Alter training rule to

$$\hat{Q}_n(s,a) \leftarrow (1-\alpha_n)\hat{Q}_{n-1}(s,a) + \alpha_n[r + \gamma \max_{a'} \hat{Q}_{n-1}(s',a')]$$

where

$$\alpha_n = \frac{1}{1 + visits_n(s,a)}$$

Can still prove convergence of $\hat{Q}$ to Q, with this and other forms of α_n [Watkins and Dayan, 1992]

Temporal Difference Learning

Q learning: reduce error between successive Q ests.

Q estimate using one-step time difference:

$$Q^{(1)}(s_t, a_t) \equiv r_t + \gamma \max_a \hat{Q}(s_{t+1}, a)$$

Why not two steps?

$$Q^{(2)}(s_t, a_t) \equiv r_t + \gamma r_{t+1} + \gamma^2 \max_a \hat{Q}(s_{t+2}, a)$$

Or n?

$$Q^{(n)}(s_t, a_t) \equiv r_t + \gamma r_{t+1} + \cdots + \gamma^{(n-1)} r_{t+n-1} + \gamma^n \max_a \hat{Q}(s_{t+n}, a)$$

Blend all of these ($0 \leq \lambda \leq 1$):

$$\begin{aligned} Q^\lambda(s_t, a_t) &\equiv (1-\lambda)\left[Q^{(1)}(s_t, a_t) + \lambda Q^{(2)}(s_t, a_t) + \lambda^2 Q^{(3)}(s_t, a_t) + \cdots\right] \\ &= r_t + \gamma \left[(1-\lambda) \max_a \hat{Q}(s_{t+1}, a) + \lambda\, Q^\lambda(s_{t+1}, a_{t+1})\right] \end{aligned}$$

TD(λ) algorithm uses above training rule

- Sometimes converges faster than Q learning
- converges for learning V^* for any $0 \leq \lambda \leq 1$ (Dayan, 1992)
- Tesauro's TD-Gammon uses this algorithm

Subtleties and Ongoing Research

- Replace $\widehat{Q}$ table with neural net (GD, EG) or other generalizer (example is $\langle s, a \rangle$, label is $\widehat{Q}(s, a)$); convergence proofs break
- Handle case where state only partially observable
- Design optimal exploration strategies
- Extend to continuous action, state
- Learn and use $\widehat{\delta} : S \times A \rightarrow S$
- Relationship to dynamic programming (can solve optimally offline if $\delta(s, a)$ & $r(s, a)$ known)
- Reinf. learning in autonomous multi-agent environments (competitive and cooperative)
  - Now must attribute credit/blame over agents as well as actions
  - Utilizes game-theoretic techniques, based on agents' protocols for interacting with environment and each other
- More info: survey papers & new textbook