Reinforcement Learning

15s1: COMP9417 Machine Learning and Data Mining

School of Computer Science and Engineering, University of New South Wales

May 27, 2015

Acknowledgements

Material derived from slides for the book
"Machine Learning" by T. Mitchell
McGraw-Hill (1997)
http://www-2.cs.cmu.edu/~tom/mlbook.html

Aims

This lecture will introduce you to reinforcement learning. Following it you should be able to:

- outline the framework of control learning
- describe control policies that choose optimal actions
- apply the method of Q learning
- outline the convergence properties of Q learning

Control Learning

Consider learning to choose actions, e.g.,

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

Control Learning

Note several problem characteristics:

- Delayed reward
- Opportunity for active exploration
- Possibility that state only partially observable
- Possible need to learn multiple tasks with same sensors/effectors

Example: TD-Gammon

Tesauro (1995)

Used Reinforcement Learning to play Backgammon

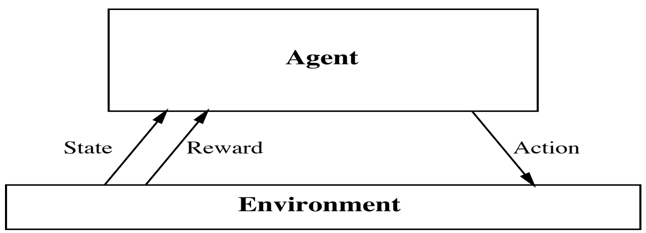
Immediate reward

- +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



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

Goal: Learn to choose actions that maximize

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

Markov Decision Processes

Assume

- finite set of states S
- set of actions A
- at each discrete time step t 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 Decision Processes

Markov assumption: $s_{t+1} = \delta(s_t, a_t)$ and $r_t = r(s_t, a_t)$

- r_t and s_{t+1} depend only on *current* 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 $\pi : S \to A$ that maximizes

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

  from any starting state in S
- where $0 \le \gamma < 1$ is the discount factor for future rewards

Note something new:

- Target function is $\pi : S \to A$
- but we have no training examples of the form $\langle s, a \rangle$
- in fact training examples are of the form $\langle \langle s, a \rangle, r \rangle$

Value Function

To begin, consider deterministic worlds ...

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

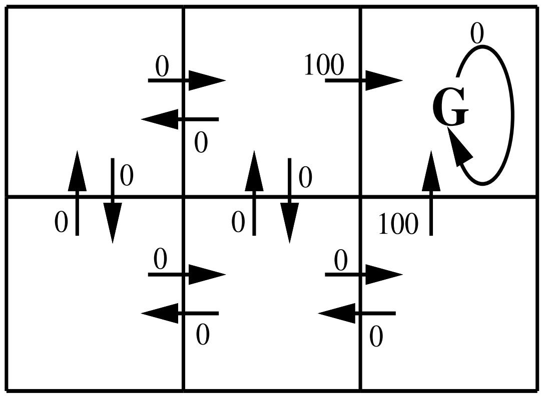
$$\begin{aligned} V^{\pi}(s) &\equiv r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots \\ &\equiv \sum_{i=0}^{\infty} \gamma^i r_{t+i} \end{aligned}$$

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^*

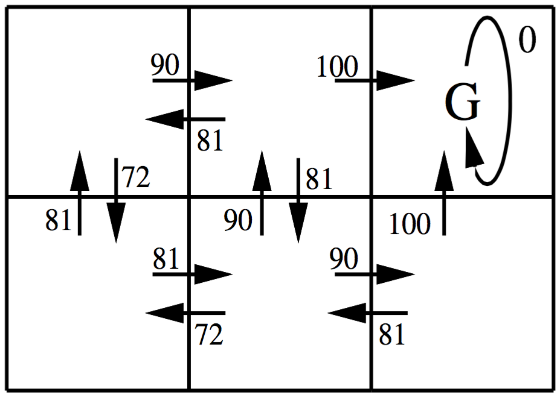
$$\pi^* \equiv \underset{\pi}{\operatorname{argmax}}\, V^{\pi}(s), (\forall s)$$

Q-learning in a simple deterministic grid world – main concepts



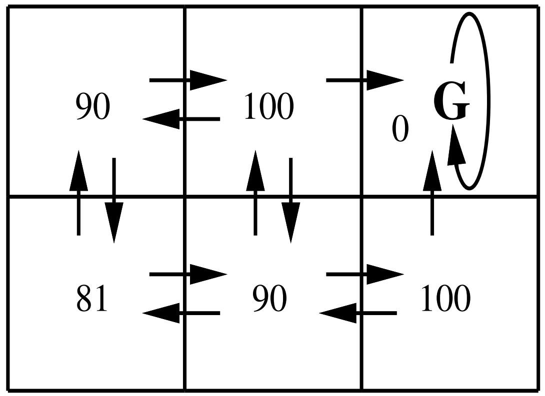
$r(s, a)$ (immediate reward) values

Q-learning in a simple deterministic grid world – main concepts



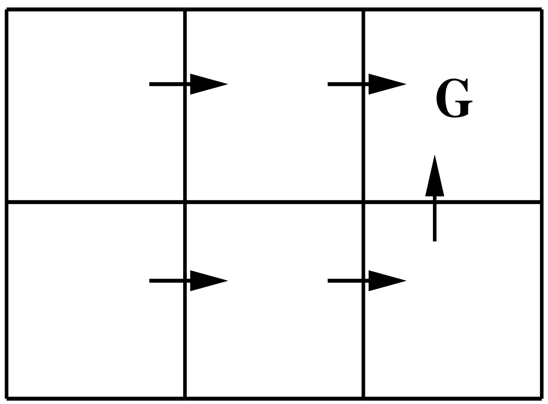
$Q(s, a)$ values

Q-learning in a simple deterministic grid world – main concepts



$V^*(s)$ values

Q-learning in a simple deterministic grid world – main concepts



One optimal policy

What to Learn

Could try to have agent learn the evaluation function V^{π^*} (which we write as V^*)

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

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

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^*

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

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

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

$$\pi^*(s) = \underset{a}{\operatorname{argmax}} Q(s,a)$$

Q is the evaluation function the agent will learn

Training Rule to Learn Q

Note that Q and V^* are 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

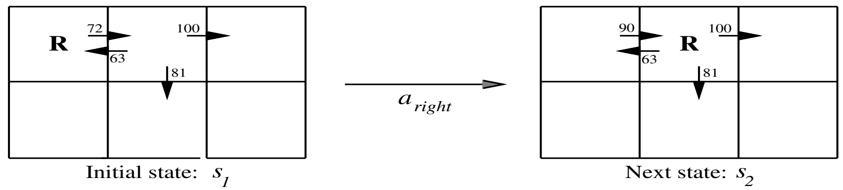
Do forever:

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

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

- $s \leftarrow s'$

Updating $\hat{Q}$



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

Note: if rewards non-negative, then

$(\forall s,a,n)\ \ \hat{Q}_{n+1}(s,a) \geq \hat{Q}_n(s,a)$ and $(\forall s,a,n)\ \ 0 \leq \hat{Q}_n(s,a) \leq Q(s,a)$

Convergence of Q learning

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

Proof: Define a full interval to be an interval during which each $\langle s, a \rangle$ is visited. 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 $\hat{Q}_n$; that is

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

Convergence of Q learning

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

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

Note we used general fact that

$$|\max_a f_1(a) - \max_a f_2(a)| \leq \max_a |f_1(a) - f_2(a)|$$

Nondeterministic Case

What if reward and next state are non-deterministic?

We redefine V, Q by taking expected values

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

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

Nondeterministic Case

Q learning generalizes to nondeterministic worlds

Alter training rule to

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

where

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

Can still prove convergence of $\hat{Q}$ to Q [Watkins and Dayan, 1992]

Temporal Difference Learning

Q learning: reduce discrepancy between successive Q estimates

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:

$$Q^\lambda(s_t, a_t) \equiv (1-\lambda)\left[Q^{(1)}(s_t, a_t) + \lambda Q^{(2)}(s_t, a_t) + \lambda^2 Q^{(3)}(s_t, a_t) + \cdots\right]$$

Temporal Difference Learning

$$Q^\lambda(s_t, a_t) \equiv (1-\lambda)\left[Q^{(1)}(s_t, a_t) + \lambda Q^{(2)}(s_t, a_t) + \lambda^2 Q^{(3)}(s_t, a_t) + \cdots\right]$$

Equivalent expression:

$$\begin{aligned} Q^\lambda(s_t, a_t) &= r_t + \gamma[\ (1-\lambda) \max_a \hat{Q}(s_t, a_t) \\ &\quad + \lambda\ Q^\lambda(s_{t+1}, a_{t+1})] \end{aligned}$$

TD(λ) algorithm uses above training rule

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

Further issues

- Replace $\hat{Q}$ table with neural net or other generalizer
- Handle case where state only partially observable
- Design optimal exploration strategies
- Extend to continuous action, state
- Learn and use $\hat{\delta} : S \times A \to S$
- Relationship to dynamic programming

Summary

- Reinforcement Learning enables learning to control (an agent)
- One of the oldest ideas in machine learning, but still useful
- Q learning is a standard approach
- Many further refinements
- Succesfully applied, e.g., in game playing
- Can take a lot of training to learn a good policy
- Exploration – exploitation trade-off (bandit learning)
- Function approximation to capture “state”
- Extended to hierarchical, relational learning