Textbook Section 17.3

Policy-iteration (S, A, P, R, γ)

For all $s \in S$ $\pi(s) \leftarrow$ random legal action;
$U(s) \leftarrow 0$

Repeat
$U \leftarrow$ Policy-Eval $(S, A, P, R, \gamma, \pi, U)$ <use π and old U to compute new U>
For all $s \in S$
If $\max_{a \in A} \sum_{s' \in S} P(s'|s,a) U(s') > \sum_{s' \in S} P(s'|s, \pi(s)) U(s')$
Then $\pi(s) \leftarrow \operatorname{argmax}_{a \in A} \sum_{s' \in S} P(s'|s,a) U(s')$

Until π doesn't change
Return π

Policy-Eval $(S, A, P, R, \gamma, \pi, U)$

For all $s \in S$ $U'(s) \leftarrow R(s) + \gamma \sum_{s' \in S} P(s'|s, \pi(s)) U(s')$

Return U'

Textbook Section 21.1 - (12.3)

What if you're not given P and R and just get info about them from acting in the world?

- Reinforcement learning

Exploration vs Exploitation
- investigate in order to learn
- stop learning and reap rewards

Key idea:
Don't learn P, R, U, or π
Learn $Q(s,a)$ - value of being in s, doing a, then acting optimally thereafter

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

$$Q(s,a) = R(s) + \gamma \sum_{s' \in S} P(s'|s,a) \max_{a'} Q(s',a')$$

↑ not known

Use instead

$$Q_{t+1}(s,a) \leftarrow Q_t(s,a) + \alpha \left([R(s) + \gamma \max_{a' \in A} Q_t(s',a')] - Q(s,a) \right)$$

↑ May depend on t, # times in s + did a

"Q Learning"

Compute $Q_t(s,a)$, $N(s,a)$ - # of times a was done in s

Initialization:

For all $s \in S$ $a \in A$ $\quad Q(s,a) \leftarrow 0$;

$N(s,a) \leftarrow 0$

Q-Learning(s,a,r,s',r') - one step update

Went from s to s' using a

$r = R(s)$, $r' = R(s')$

$N(s,a) \leftarrow N(s,a) + 1$

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left(r + \max_{a' \in A} Q(s',a') - Q(s,a) \right)$$

Return Next action to take

$$\operatorname{argmax}_{a' \in A} f(Q(s',a'), N(s',a'))$$

↑ implement exploitation vs exploration

Example $f(u,n) = \begin{cases} R_{max} & \text{if } n < N \\ u & \text{otherwise} \end{cases}$

In a given state all actions give value R_{max} until you've seen that action in that state N times

Exam Topics

Problem space search (DFS, BFS, A*, etc.)
Game Tree Search
Propositional logic (incl. esp. resolution)
First order logic (incl. resolution, unification, occurs check, etc.)
Perceptrons
Backprop
Reinforcement learning (incl. value iteration, policy iteration, Q learning)