Policy-iteration \((S, A, P, R, \gamma)\)

For all \(s \in S\) \(\Pi(s) \in \text{random legal action;}
U(s) \leftarrow 0\)

Repeat

\[
U \leftarrow \text{Policy-Eval} (s, A, P, R, \gamma, \Pi(s), U) \quad \text{<use } \Pi \text{ and old } U \text{ to compute new } U>
\]

For all \(s \in S\)

If \[
\max_{a \in A} \sum_{s' \in S} \gamma \cdot P(s' \mid s, a) U(s') > \sum_{s' \in S} \gamma \cdot P(s' \mid s, \Pi(s)) U(s')
\]

Then \(\Pi(s) \leftarrow \arg \max_{a \in A} \sum_{s' \in S} \gamma \cdot P(s' \mid s, a) U(s')\)

Until \(\Pi\) doesn't change

Return \(\Pi\)

Policy-Eval \((s, A, P, R, \gamma, \Pi(s), U)\)

For all \(s \in S\)

\[
U'(s) \leftarrow R(s) + \gamma \sum_{s' \in S} \gamma \cdot P(s' \mid s, \Pi(s)) U(s')
\]

Return \(U'\)

Textbook Section 17.3

Textbook Section 21.1-(12.3)

what if you're not given \(P\) and \(R\) and just set 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 \(P, R, U, \text{ or } \Pi\)

Learn \(Q(s,a)\) - value of being in \(s\), doing \(a\),
then acting optimally thereafter

\[
\Pi^*(s) = \arg \max_{a \in A} Q(s,a)
\]
\[ Q(s, a) = R(s) + \gamma \sum_{s' \in S} P(s' | s, a) \max_{a'} Q(s', a') \]

not known

Use instead

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

May depend on

\( t, \# \text{times in } s \)

+ d.d. o

"Q-Learning"

**Initialization**

For all \( s \in S, a \in A \)

\[ Q(s, a) = 0; \]

\[ N(s, a) = 0 \]

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

\[ \text{Visit from } s \text{ to } s' \text{ using } a \]

\[ r = R(s), r' = R(s') \]

\[ N(s, a) = N(s, a) + 1 \]

\[ Q(s, a) = Q(s, a) + \alpha \left( r + \max_{a' \in A} Q(s', a') - Q(s, a) \right) \]

Return next action to take

\[ \text{argmax } f(Q(s', a'), N(s', a')) \]

implement exploitation vs exploration

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

In a given state all actions give value \( R_{\text{max}} \)

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.)
Perceptions
Backprop
Reinforcement learning (incl. value iteration, policy iteration, Q learning)