Reinforcement learning

* What if \( P(s'|s,a) \) and \( R(s) \) are not known?
  Can we learn \( T^* \) or \( V^*(s) \) from experience?

1) Model-based (indirect) approach

Explore world, estimate model \( \hat{P}(s'|s,a) \approx P(s'|s,a) \), compute \( \hat{T}^* \) from \( \hat{F}(s'|s,a) \)

* Cons: to store \( P(s'|s,a) \) is \( O(n^2) \) for \( n \) states
  Only care about \( T^*(s) \) or \( V^*(s) \) which are \( O(n) \)
  Is it really necessary to estimate a model?

* Pro: model \( P(s'|s,a) \) useful for task transfer, where rewards \( R(s) \) or discount factor \( \gamma \) changes, but \( P(s'|s,a) \) stays the same

2) Direct approach: learn \( T^*(s) \), \( V^*(s) \) w/o building model. How?

Stochastic approximation theory

* How to estimate mean of random variable \( X \) from samples \( X_0, X_1, \ldots, X_t \)?

1) obvious sample average

\[
\mu = \frac{1}{t} (X_0 + X_1 + \ldots + X_t)
\]

: estimate converges to mean \( \mu \rightarrow E[X] \) as \( T \rightarrow \infty \) by law of large numbers.

2) incremental update

initialize \( \mu_0 = 0 \)

update \( \mu_t = \left(1 - \alpha_t\right) \mu_{t-1} + \alpha_t X_t \) for \( 0 < \alpha_t < 1 \)

also write this as: \( \mu_t = \mu_{t-1} + \alpha_t (X_t - \mu_{t-1}) \)

known as TD learning algorithm.

Thm: \( \mu_t \rightarrow E[X] \) as \( t \rightarrow \infty \) with probability 1 if

\[\sum_{t=1}^{\infty} \alpha_t = \infty \] (diverges)

\[\sum_{t=1}^{\infty} \alpha_t^2 < \infty \] (converges)

Intuitively:

(i) \( \alpha_t \) decays sufficiently slowly to incorporate large # samples.

(ii) \( \alpha_t \) decays sufficiently fast to allow for convergence (damp oscillations).
**Temporal difference (TD) prediction**

* How to evaluate policy without model?
  How to compute $V^\pi(s)$ without knowing $P(s' | s, \pi(s))$?

* Explore state space via policy $\pi$

  $\begin{align*}
  \text{actions} & \quad T_i(s_0) \quad T_i(s_1) \\
  \text{states} & \quad S_0 \quad S_1 \quad \rightarrow \\
  \text{rewards} & \quad R(s_0) \quad R(s_1)
  \end{align*}$

* Recall Bellman equation:

  $V^\pi(s) = R(s) + \gamma \sum_{s'} P(s' | s, \pi(s)) V^\pi(s')$

* TD learning algorithm

  Initialize $V_{t+1}(s) = 0$ for all $s$ (at time $t=0$)

  Update: $V_{t+1}(s_{t+1}) = V_t(s_t) + \alpha \left[ R(s_{t+1}) + \gamma V_t(s_{t+2}) - V_t(s_t) \right]$.

* Features:
  
  - Update after each step of experience.
  - Learns directly from experience w/o model.
  - Easy to implement.

* Asymptotic convergence

  $\lim_{t \to \infty} V_t(s) \longrightarrow V^\pi(s)$?

  Assume that each state of MDP is visited infinitely often by policy $\pi$.

  Then, TD($\phi$) converges:
  
  - "with probability 1" if:
    
    - each state $s$ has its own learning rate $\alpha_v(s)$ where $V$ denotes
      
      # visits so far to state $s$
    
    - learning rates satisfy for all states $s$.

      (i) $\lim_{t \to \infty} \alpha_v(s) = \infty$

      (ii) $\lim_{t \to \infty} \alpha_v^2(s) < \infty$

    Should agents in practice enforce (i) and (ii)?

    - Yes, for theoretical convergence guarantee
    - No, for non-stationary worlds where MDP is just an approximation.

    - "in mean" if step size $\alpha$ is constant and sufficiently small.
Q-learning

* How to optimize policy \( \Pi^* \) without model \( P(s' | s, a) \)?
* How to compute \( Q^* (s, a) \) without model?
* Explore state-action space at random:
  actions \( a_0 \) \( a_1 \) ... Not following any particular policy!
  states \( s_0 \) \( s_1 \)
  rewards \( R(s_0) \) \( R(s_1) \)

* Bellman optimality equation
  \[
  Q^* (s, a) = R(s) + \gamma \sum_{s'} P(s' | s, a) V^*(s')
  \]

* One-step Q-learning:
  Initialize \( Q_0 (s, a) = 0 \) for all states \( s \) and actions \( a \).
  \[
  Q_{t+1} (s, a) = Q_t (s, a) + \alpha [ R(s_t) + \gamma \max_{a'} Q_t (s_{t+1}, a') - Q_t (s_t, a) ]
  \]

* Features:
  - simple, incremental
  - model-free
  - experience-based

* Asymptotic convergence: \( \lim_{t \to \infty} Q_t (s, a) \to Q^* (s, a) \) appropriately

Thm: if each state-action pair is visited infinitely often, and a decreasing step size \( \alpha(t) \) is used for each state-action pair, then Q-learning converges with probability one.