Computational Cognitive Modeling
HW 2 Review

03/10/2020

RL Methods

I. Policy Iteration: Evaluation, Improvement

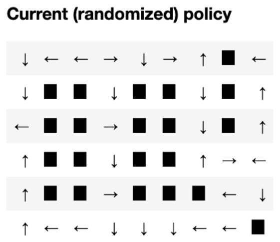
II. Monte Carlo Methods

III. Temporal Difference Learning

i. Dynamic Programming: Policy Iteration

- Two stages

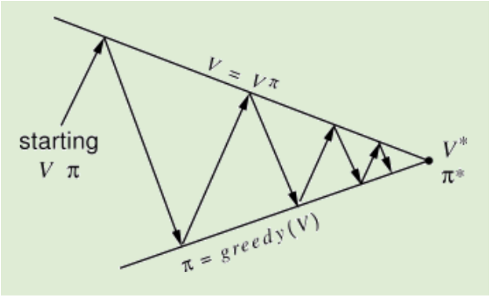
π



- Policy Evaluation: assign value to states
- Uses your current policy to get values

Bellman equation

$$V^{\pi}(s) = \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'} [\mathcal{R}^a_{ss'} + \gamma V^{\pi}(s')]$$



- Policy Improvement: use values to choose actions
  - Uses values of states to choose actions
  - Q-values: state-action pairs

$$\begin{aligned} q_\pi(s,a) &\doteq \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = a] \\ &= \sum_{s',r} p(s',r \mid s,a)\Big[r + \gamma v_\pi(s')\Big]. \end{aligned}$$

i. Policy Iteration

look at Problem 1 code*

RL Methods

I. Policy Iteration: Evaluation, Improvement

II. Monte Carlo Methods

III. Temporal Difference Learning

ii. First-visit Monte Carlo

- Each time you *initially* visit a state, record the rewards you receive until the end of an episode

- Sample *many episodes*, then average the rewards received for each state
  - This gives you an estimate of the value of the state or action

- Use this estimate to update your policy (greedy method)

ii. First-visit Monte Carlo

look at Problem 2 code*

Especially the pseudo-code explanation

RL Methods

I. Policy Iteration: Evaluation, Improvement

II. Monte Carlo Methods

III. Temporal Difference Learning

iiii. Temporal Difference

- Incremental updating: Q-learning update rule
  - "Off-policy": Q-value based on max in the next state

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\Big].$$

iiii. Temporal Difference

- Incremental updating: Q-learning update rule
  - "Off-policy": Q-value based on max in the next state

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \Big[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \Big].$$

This differs from SARSA (on-policy):

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \Big[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t) \Big].$$

iiii. Temporal Difference

- Incremental updating: Q-learning update rule
  - “Off-policy”: Q-value based on max in the next state

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\Big].$$

- For HW Problem 7: adopt some code from your monte-carlo solution
  - Previously computed discounted average returns from an episode
  - Here: asked to *incrementally* update the Q-value
    - While balancing explore/exploit

iii. Temporal Difference

look at Problem 7 code*

Questions?